The Dissertation Committee for Peng Yu
certifies that this is the approved version of the following dissertation:

Fast and Accurate Lithography Simulation and Optical Proximity Correction for Nanometer Design for Manufacturing

Committee:

David Z Pan, Supervisor

Jacob A Abraham

Sanjay K Banerjee

Chris A Mack

C Grant Willson

Fast and Accurate Lithography Simulation and Optical Proximity Correction for Nanometer Design for Manufacturing

by

Peng Yu, B.S.; M.S.

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

May 2009

Acknowledgments

I would like to thank my advisor, Dr. David Z. Pan, for his guidance, encouragement and support over the years. The freedom he gave me in working on my dissertation was very valuable and I learned a lot in this process. I am grateful to Dr. Chris A. Mack who introduced me to the field of lithography simulation and has given me many insights on this subject. I also would like to thank Dr. Craig M. Chase from who I have learned many C++ programmning skills, without which this dissertation would not be possible. Many thanks also to my PhD committee members, Dr. Jacob A. Abraham, Dr. Sanjay K. Banerjee and Dr. C. Grant Willson.

My time at the Univeristy of Texas was an enjoyable one, largely thanks to the positive and cooperative atmosphere of Dr. Pan's research group. I would like to thank Danhua Shao from the Department of Electrical and Computer Engineering, Weifeng Qiu, Wenhao Wang and Henry Chang from the Institute for Computational Engineering Sciences, and Wing-chi Poon and Yancong Zhou from the Department of Computer Science for their friendship and for many interesting and entertaining discussions. Special thanks to Andrew Kieschnick, who has given me generous technical assistance. I also would like to thank Kathlean Rice for proofreading this dissertation.

Finally, I would like to thank my parents, to whom this dissertation is dedicated, for the encouragement they have alway given to me.

Fast and Accurate Lithography Simulation and Optical Proximity Correction for Nanometer Design for Manufacturing

Publication No. _____

Peng Yu, Ph.D.
The University of Texas at Austin, 2009

Supervisor: David Z Pan

As semiconductor manufacture feature sizes scale into the nanometer dimension, circuit layout printability is significantly reduced due to the fundamental limit of lithography systems. This dissertation studies related research topics in lithography simulation and optical proximity correction.

A recursive integration method is used to reduce the errors in transmission cross coefficient (TCC), which is an important factor in the Hopkins Equation in aerial image simulation. The runtime is further reduced, without increasing the errors, by using the fact that TCC is usually computed on uniform grids. A flexible software framework, ELIAS, is also provided, which can be used to compute TCC for various lithography settings, such as different illuminations.

Optimal coherent approximations (OCAs), which are used for full-chip image simulation, can be speeded up by considering the symmetric properties

of lithography systems. The runtime improvement can be doubled without loss of accuracy. This improvement is applicable to vectorial imaging models as well. Even in the case where the symmetric properties do not hold strictly, the new method can be generalized such that it could still be faster than the old method.

Besides new numerical image simulation algorithms, variations in lithography systems are also modeled. A Variational LIthography Model (VLIM) as well as its calibration method are provided. The Variational Edge Placement Error (V-EPE) metrics, which is an improvement of the original Edge Placement Error (EPE) metrics, is introduced based on the model. A *true* process-variation aware OPC (PV-OPC) framework is proposed using the V-EPE metric. Due to the analytical nature of VLIM, our PV-OPC is only about 2-3× slower than the conventional OPC, but it *explicitly* considers the two main sources of process variations (exposure dose and focus variations) during OPC.

The EPE metrics have been used in conventional OPC algorithms, but it requires many intensity simulations and takes the majority of the OPC runtime. By making the OPC algorithm intensity based (IB-OPC) rather than EPE based, we can reduce the number of intensity simulations and hence reduce the OPC runtime. An efficient intensity derivative computation method is also provided, which makes the new algorithm converge faster than the EPE based algorithm. Our experimental results show a runtime speedup of more than 10× with comparable result quality compared to the EPE based OPC.

The above mentioned OPC algorithms are vector based. Another categories of OPC algorithms are pixel based. Vector based algorithms in general generate less complex masks than those of pixel based ones. But pixel based

algorithms produce much better results than vector based ones in terms of contour fidelity. Observing that vector based algorithms preserve mask shape topologies, which leads to lower mask complexities, we combine the strengths of both categories—the topology invariant property and the pixel based mask representation. A topological invariant pixel based OPC (TIP-OPC) algorithm is proposed, with lithography friendly mask topological invariant operations and an efficient Fast Fourier Transform (FFT) based cost function sensitivity computation. The experimental results show that TIP-OPC can achieve much better post-OPC contours compared with vector based OPC while maintaining the mask shape topologies.

Table of Contents

List of Tables

List of Figures

Chapter 1

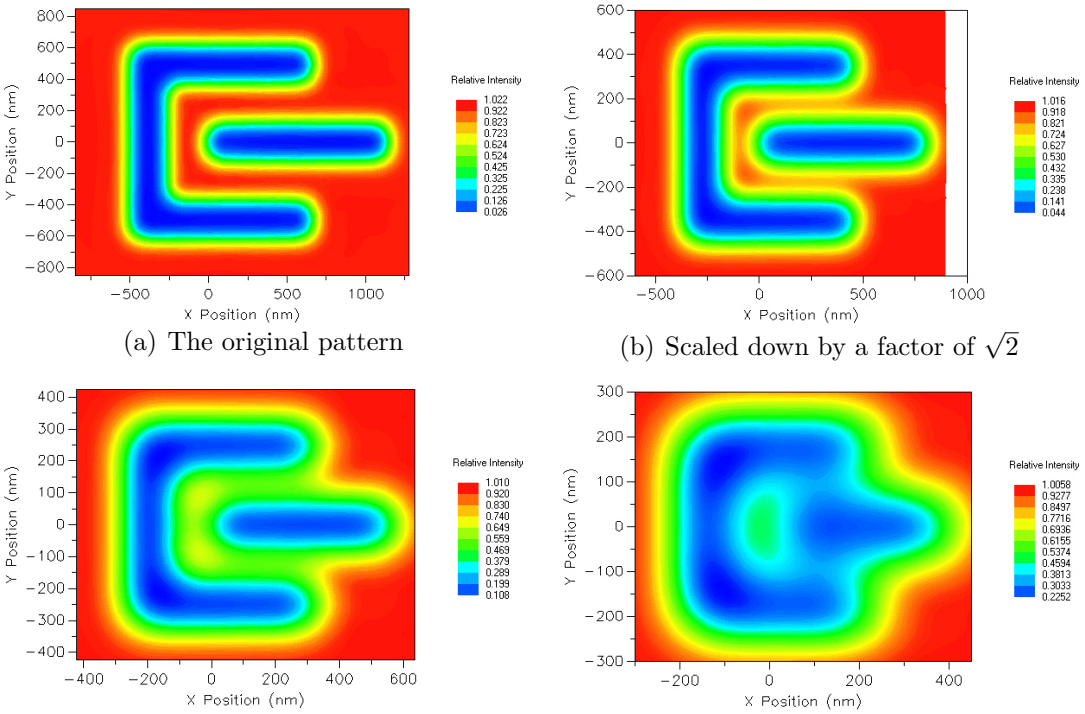
Introduction

In this chapter, the motivation of this dissertation will be presented. Lithography simulation, which is the basis of this dissertation, will also be discussed.

1.1 Motivation

Lithography is widely used in today's semiconductor industry to print circuit layout patterns. Due to the continuous miniaturization of feature sizes, layout pattern printability and process window are significantly reduced due to the fundamental limit of the microlithography systems and process variations. The feature sizes have been scaled much faster than the lithography wavelength. For example, according to International Technology Roadmap for Semiconductors (ITRS), Microprocessor Unit (MPU) physical gate length will be 10 nm by the year 2015. It is also predicted that 193 nm immersion lithography may still be used at that time. Figure 1.1 shows how a print pattern is deteriorated as the layout pattern is scaled down. These pose new challenges for the accuracy and speed lithography simulation and Optical Proximity Correction (OPC) [24].

The fundamental limits of a lithography system, summarized in [12, 58, 59, 97], refer to the achievable lithography system bounds, e.g., resolution in



(a) The original pattern

(b) Scaled down by a factor of $\sqrt{2}$

(c) Scaled down by a factor of 2

(d) Scaled down by a factor of $2\sqrt{2}$

Figure 1.1: The printed patterns become more deviated from the mask pattern as the mask pattern scales down. The original mask pattern looks like the pattern in (a). Simulated using PROLITH$^{\text{TM}}$ with 193 nm wavelength, conventional partially coherent illumination ($\sigma = 0.8$), NA $= 0.7$.

terms of pitch and critical dimension, as shown in Figure 1.2. Figure 1.3 shows

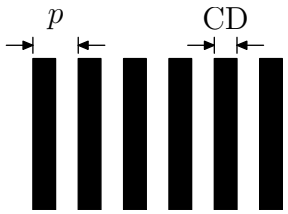


Figure 1.2: A periodic line/space pattern. The pitch p denotes its period in space. Critical Dimension (CD) refers to the line width in this case.

a sketch of a typical lithography system. The mask is illuminated by the light source through the illumination lens. An image of the mask is formed in the photoresist through the projection lens. The wavelength of the light source is

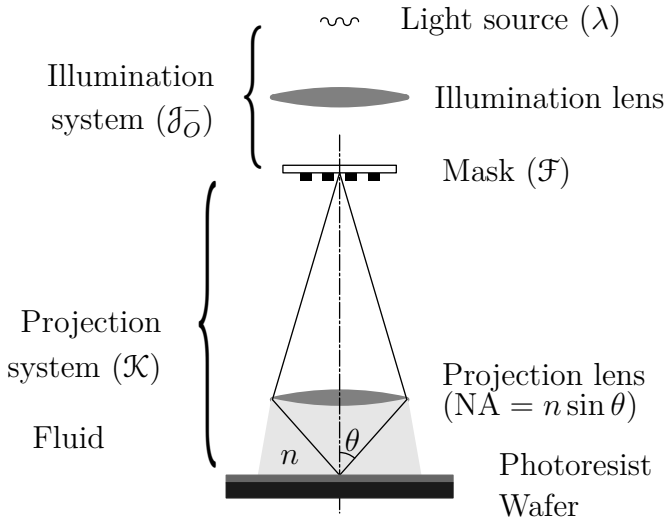


Figure 1.3: A typical lithography system. The mask is illuminated by a laser light source with wavelength λ. The photoresist is exposed though the projection system. The latent image is formed in the photoresist.

denoted as λ. The index of refraction of the medium between the lens and the photoresist is denoted as n. For 193nm water immersion lithography, $n = 1.44$;

for air, $n \approx 1$. θ is the half-angle of the maximum cone of light that exits the lens.

The optimal illumination scheme for the line/space pattern is a dipole. The minimum printable pitch using this illumination is given by

$$p_{\min} = \frac{\lambda}{2n \sin\theta}, \tag{1.1}$$

where $n \sin\theta$ is defined as the *numerical aperture* (NA). In general, for any pattern (including line/space), the minimum pitch, in a broader sense, has the following form

$$p_{\min} = k_1 \frac{\lambda}{n \sin\theta}, \tag{1.2}$$

where k_1 is an illumination and pattern dependent factor, and its physical minimum is 1/2.

Depth-of-Focus (DOF), indicating the pattern robustness with respect to focus variation, is defined as the range of focus that keeps the resulting printed feature within a variety of specifications (such as line width, sidewall angle, photoresist loss, and exposure latitude). It is estimated as

$$\text{DOF} = \frac{n\lambda}{2(1 - \cos\theta)}. \tag{1.3}$$

To print smaller pitch at a given wavelength λ, we can increase θ and n. However, unless higher n material is available to replace water, DOF will continue to decrease as θ goes to its physical limit 90°. Therefore, more CD variations are expected due to focus variations. Meanwhile, the exposure dose variation impact will also become more severe.

Because of the increasing difficulties in lithography manufacture process, faster and more accurate lithography simulation-based applications play

a more important role in the semiconductor industry [35]. They include, but are not limited to, OPC, Resolution Enhancement Techniques (RET) [98], post-OPC silicon image verification and design rule definition. Since all these applications are based on lithography simulation, they are called Computational Lithography (CL) [6, 96].

The industry has pushed hard to reduce simulation runtime and to improve simulation accuracy by using parallel computation and dedicated hardware. Mentor Graphics has used multiprocessing and multithreading on Linux workstation clusters [17]. Specific hardware-accelerated computational lithography platform has also been used in the industry [102]. IBM has developed software for IC design and DFM software with the IBM's BlueGene supercomputer [36]. These efforts are very important but require huge software and hardware investments. In this dissertation, we will make improvements in lithography simulation and OPC in the algorithmic aspects.

1.2 Lithography Modeling

Various models have been developed for lithography system simulation. Based on the details of the physics descriptions, these models in general can be classified into two categories — physics based models and phenomenological models [25, 26, 80, 84].

The physics based model can be divided into three steps:

- Photomask patterns are transformed to a chemical latent image in the photoresist bulk through the optical system by exposing the photomask.
- The chemical latent image is diffused in the post exposure bake (PEB) step. Chemical reactions take place in the photoresist.

- The photoresist development results in 3-dimensional photoresist profiles.

Usually, the physics based models are slower but more accurate than the phenomenological ones. Because of the modeling of fundamental physics and chemistry in lithography systems, physics based models (eg. PROLITH$^{\text{TM}}$ and Solid-E$^{\text{TM}}$) can tell the consequences of the process parameter alternations, which help the process engineers to fine tune the lithography processes. However, it is difficult to calibrate these models because of their complexities and the difficulties in measuring the model parameters. Phenomenological models, on the other hand, seek for simulation speed and reasonable accuracy. They do not model all the physics and chemistry in the lithography and only work in a very limited domain of the process parameter space. But, it is relatively easy to fit them to the experiments because of their simplicities.

OPC requires fullchip lithography simulation, which must be fast and reasonably accurate. The phenomenological models are the best candidates. We review the phenomenological lithography modeling in the following subsections, which roughly correspond to the three steps of the physics based model.

1.2.1 Imaging Basics

An aerial image, by definition, is a projected image which is "floating in air." In lithography, it usually refers to the image on top of the photoresist or in some plane in the photoresist bulk. It is described by the *Hopkins Equation* [7, 49]

$$\mathcal{I}(\boldsymbol{k}) = \iint_{-\infty}^{+\infty} \mathcal{T}(\boldsymbol{k}+\boldsymbol{k}', \boldsymbol{k}')\mathcal{F}(\boldsymbol{k}+\boldsymbol{k}')\mathcal{F}^*(\boldsymbol{k}')\mathrm{d}^2\boldsymbol{k}'. \tag{1.4}$$

$\mathcal{F}(\boldsymbol{k})$ is the mask transmission function $F(\boldsymbol{r})$ in the frequency domain, where $\boldsymbol{k}$ denotes a point in the frequency domain and $\boldsymbol{r}$ denotes a point in the spatial domain. The superscript * denotes the complex conjugation operation. $\mathcal{I}(\boldsymbol{k})$ is the image intensity in the frequency domain. $\mathcal{T}(\boldsymbol{k}, \boldsymbol{k}')$ is the *transmission cross coefficient* (TCC), given by

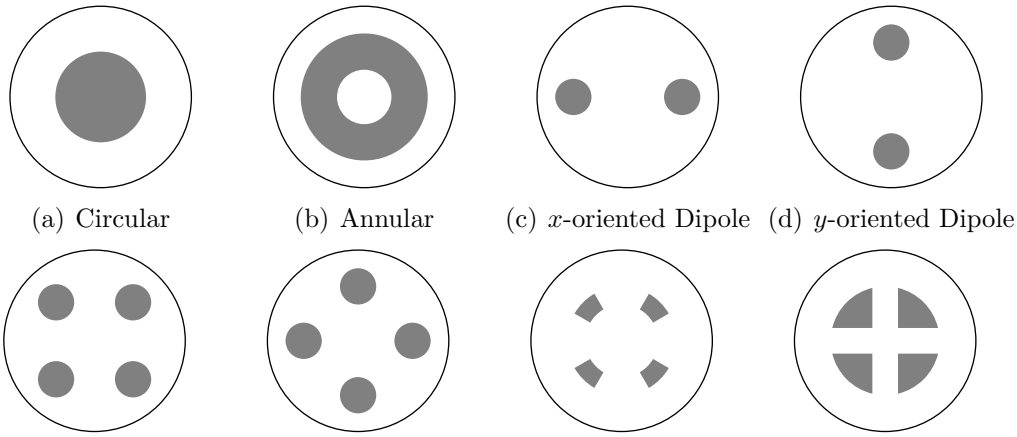
$$\mathcal{T}(\boldsymbol{k}', \boldsymbol{k}'') = \iint_{-\infty}^{+\infty} \mathcal{J}(\boldsymbol{k}) \mathcal{K}(\boldsymbol{k} + \boldsymbol{k}') \mathcal{K}^*(\boldsymbol{k} + \boldsymbol{k}'') \mathrm{d}^2\boldsymbol{k}. \tag{1.5}$$

The meanings of the symbols in (1.5) are described below:

- $\mathcal{J}(\boldsymbol{k})$ is the illumination function, which satisfies

$$\mathcal{J}(\boldsymbol{k}) = 0, \quad \text{for } k \geqslant 1, \tag{1.6}$$

  where $k = |\boldsymbol{k}|$. We illustrate some commonly used ones in Figure 1.4.



(a) Circular (b) Annular (c) x-oriented Dipole (d) y-oriented Dipole

(e) Normal Quadrupole (f) Cross Quadrupole (g) Quasar (h) Shrinc

Figure 1.4: Some commonly used illumination schemes. The outer circles are references, whose radii are all 1. $\mathcal{J}$ is a constant over the gray regions.

- $\mathcal{K}(\boldsymbol{k})$ is the projection system transfer function. It can be written as

$$\mathcal{K}(\boldsymbol{k}) = e^{i2\pi(z\phi(\boldsymbol{k})+\Phi(\boldsymbol{k}))}\mathcal{K}_0(\boldsymbol{k}), \tag{1.7}$$

where $\phi(\boldsymbol{k}) = \sqrt{1-k^2\sin^2\theta_{\text{obj}}}$, θ_{obj} is the semi-aperture angle at the image plane [99], z is the focus error scaled by the wavelength λ and $\Phi(\boldsymbol{k})$ denotes the aberration term. Assuming a circular pupil, $\mathcal{K}_0(\boldsymbol{k})$ can be written as

$$\mathcal{K}_0(\boldsymbol{k}) = \begin{cases} 1 & k < 1 \\ 0 & \text{otherwise} \end{cases} . \tag{1.8}$$

1.2.2 Resist Blur

The diffusion process is modeled by convolving the pre-PEB latent image with a function called the blur function. A number of blur functions have been proposed [4, 11, 30, 32–34]. We use a Gaussian blur function

$$G(\boldsymbol{x}) = \frac{1}{\sqrt{2\pi}d}e^{-\frac{x^2}{2d^2}}, \tag{1.9}$$

where d is the diffusion length and $\boldsymbol{x}$ denotes a spatial point. The Fourier transform of this function is

$$\mathcal{G}(\boldsymbol{k}) = e^{-2\pi^2 d^2 k^2}, \tag{1.10}$$

where $k = |\boldsymbol{k}|$. The TCC for latent images in photoresist is given by

$$\mathcal{T}_G(\boldsymbol{k}', \boldsymbol{k}'') = \mathcal{G}(\boldsymbol{k}' - \boldsymbol{k}'') \iint \mathcal{J}(\boldsymbol{k})\mathcal{K}(\boldsymbol{k}+\boldsymbol{k}')\mathcal{K}^*(\boldsymbol{k}+\boldsymbol{k}'')\mathrm{d}^2\boldsymbol{k}. \tag{1.11}$$

The latent images can be computed by Hopkins Equation as well

$$\mathcal{I}_G(\boldsymbol{k}) = \iint_{-\infty}^{+\infty} \mathcal{T}_G(\boldsymbol{k}+\boldsymbol{k}', \boldsymbol{k}')\mathcal{F}(\boldsymbol{k}+\boldsymbol{k}')\mathcal{F}^*(\boldsymbol{k}')\mathrm{d}^2\boldsymbol{k}'.$$

1.2.3 Photoresist Modeling

The photoresist development model is used to predict the print contour based on the latent image. To make the simulation fast, these models usually are empirical.

One commonly used such model is the threshold bias photoresist model. It has been demonstrated in [33] that it predicts CD fairly accurately. This model assumes the printed contour can be computed by applying a constant bias (B) to the contour where the intensity is equal to an intensity threshold I_{th} (Figure 1.5). B and I_{th} are parameters subject to calibration (see Section 4.3.2).

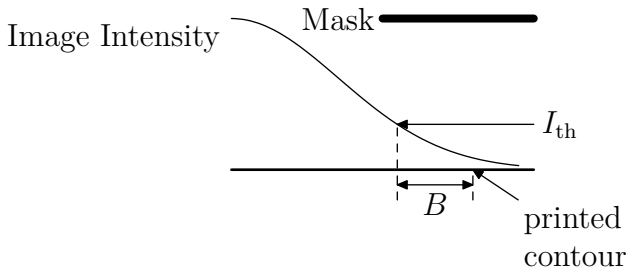


Figure 1.5: Threshold bias model. The image intensity is denoted by the curve. A constant bias B is applied the location where the intensity equals to the intensity threshold I_{th} to get the printed contour.

Another commonly used photoresist model is the variable threshold model [41, 76]. In this model, the printed contour is also determined by a threshold. The threshold is determined based on some empirical formula of a number of image quantities, such as the maximum intensity, the minimum intensity, and the intensity slope. The empirical formula is also subjected to calibration.

1.3 Optimal Coherent Approximations

Although the Hopkins Equation can be directly used to computed lithography images, it is too slow to be used for large scale lithography simulation. Instead, the Optimal Coherent Approximations (OCAs) [67, 68] are used, which are much faster than the Hopkins Equation.

Based on OCAs, it can be derived from the Hopkins Equation that the images can be computed as

$$I(\boldsymbol{r}) = \sum_{n=0}^{P-1} \sigma_n \Big| Q_n \ast\ast F \Big|^2, \tag{1.12}$$

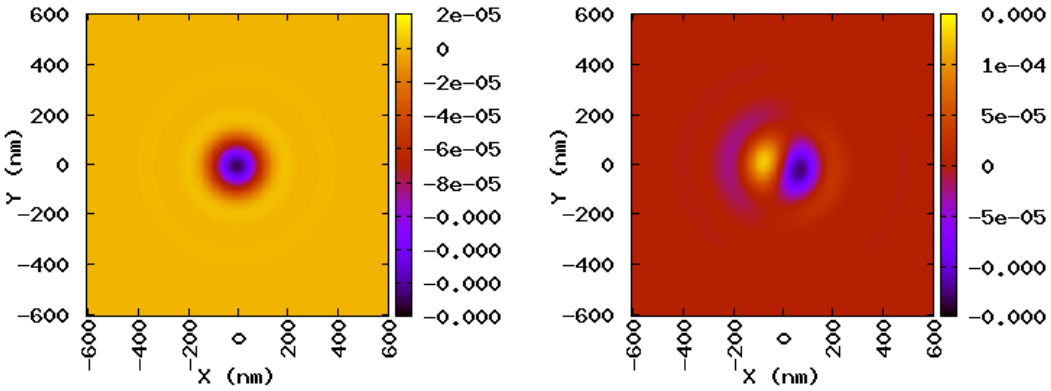
where P is a constant and $\ast\ast$ is the convolution operator and Q_n's (called kernels) are complex functions. Here, the kernels can be computed by the method shown in [24]. Figure 1.6 shows the first six kernels for a lithography system with the conventional partially coherent illumination $\sigma = 0.7$, the numerical aperture NA $= 0.8$ and the wavelength $\lambda = 193\,\text{nm}$. In general, the kernels are almost zero outside a region of the size about a few microns. Therefore, to compute the convolution value at a point, we only need the mask shapes in the region, which is called the support region.

For commonly used masks, such as binary mask (BIM) or phase shift mask (PSM) with the phases of 0° and 180°, the mask transmission function $F(\boldsymbol{r})$ is real. Therefore, by separating Q_n's real and imaginary parts, (1.12) can be written as

$$I(\boldsymbol{r}) = \sum_{n=0}^{P-1} \sigma_n \Big((\Re Q_n \ast\ast F)^2 + (\Im Q_n \ast\ast F)^2 \Big).$$

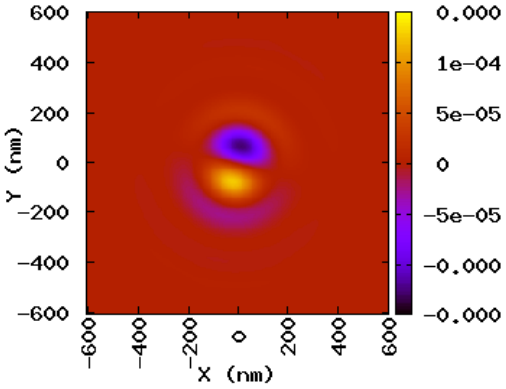
Renaming σ_n, $\Re Q_n$ and $\Im Q_n$ by

$$\sigma_0, \sigma_0, \cdots, \sigma_{P-1}, \sigma_{P-1} \to \sigma_0, \cdots, \sigma_{2P-1},$$
$$\Re Q_0, \Im Q_0, \cdots, \Re Q_{P-1}, \Im Q_{P-1} \to Q_0, \cdots, Q_{2P-1},$$

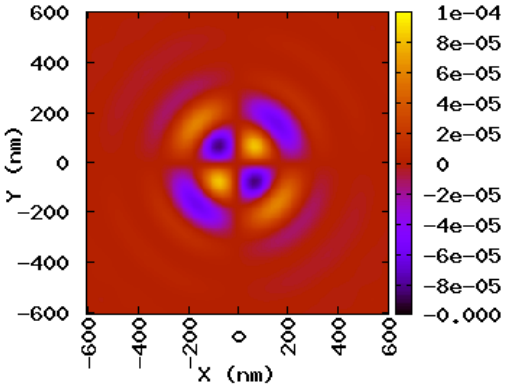


(a) Q_0

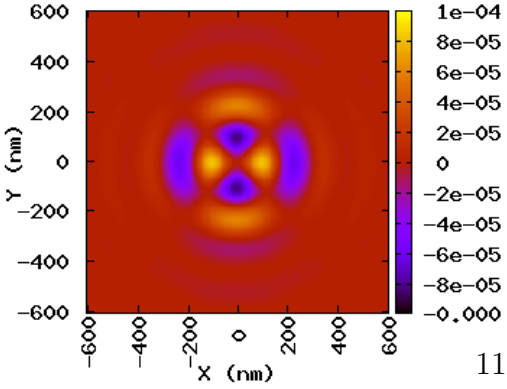
(b) Q_1



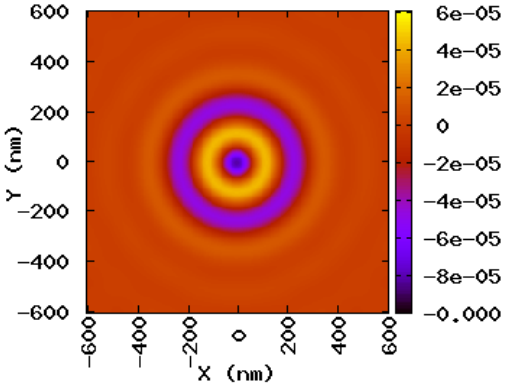
(c) Q_2



(d) Q_3



(e) Q_4



(f) Q_5

Figure 1.6: The first six kernels (conventional partially coherent illumination $\sigma = 0.7$, NA $= 0.8$, $\lambda = 193\,\text{nm}$).

we have a different form

$$I(\boldsymbol{r}) = \sum_{n=0}^{2P-1} \sigma_n (Q_n \ast\ast F)^2. \tag{1.13}$$

Therefore, an image can be computed by convolving a set of real kernels with the mask.

1.4 Summary of a Typical Lithography Simulation and OPC Flow

In summary, Figure 1.7 shows a typical fullchip lithography simulation and OPC flow. The transmission cross coefficient (TCC) is first computed

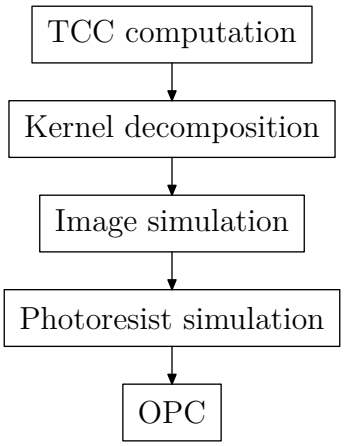


Figure 1.7: A typical fullchip lithography simulation and OPC flow.

based on the optics information. It can be decomposed into a set of kernels using Optimal Coherent Approximations (OCAs). The images can be simulated by convolving masks with the kernels. The final photoresist profile can then be simulated.

Based on lithography simulations, OPC modifies mask patterns to improve the image printability and the image robustness. OPC algorithms can be

classified as *sparse* OPC and *dense* OPC [15, 16, 19]. These two categories are mainly different in computing the photoresist profile either on sparsely sampled sites or on dense grids, as well as in treating mask geometries as polygons or pixel-based images. Sparse OPC is the current dominant OPC methodology, while the dense OPC is gaining more interest. Cobb did pioneering work on sparse OPC [20, 21, 24]. Granik et al. formulated OPC problem into a more rigorous framework based on Mask Error Enhancement Factor (MEEF) theory [22, 43]. In this dissertation, we will make improvement on TCC computation (Chapter 2), kernel decomposition (Chapter 3) and OPC (Chapter 4, 5 and 6).

Chapter 2

Transmission Cross Coefficient Computation with Accurate Numerical Algorithms

In this chapter, we demonstrate that the boundaries of the support of the integrand of TCC introduces the majority of the numerical errors. The error can be reduced dramatically by using the recursive integration method. Because TCC is often computed on uniform grids, the algorithm can be further speeded up without increasing the errors. Given the same numerical accuracy, the experiments show that our new algorithm can speed up the runtime by thousands of times. The very accurate results that are computed in a reasonable amount of time can be used to benchmark other lithography aerial simulators. We also develop a flexible software framework called ELIAS [105] so that the users of ELIAS can easily use other lithography settings.

Preliminary results of this work have been published in [108].

2.1 Introduction

As feature size gets smaller, smaller simulation errors are required. More lithography simulation benchmarking methods have been proposed [45, 81–83, 89]. In this chapter, we propose a new benchmarking method that can be used for arbitrary lithography settings, while previous methods only benchmark some specific cases that have analytical solutions [83].

We prove that the jump-discontinuity of the integrand of TCC on the boundary of the integrand's support is the major source of TCC errors. We improve the computation accuracy by specially integrating the discontinuous regions using a recursive integration method. The flow in Figure 1.7 requires the computation of the function values of TCC on a uniform grid, which form a 4-dimensional TCC matrix. By taking advantage of the correction between the entries within a TCC matrix, we can speed up its computation without losing accuracy. The increase in the accuracy of TCC in turn increases the accuracy of the aerial image simulation.

We implemented the algorithm in our software package ELIAS written in C++. It can be extended to support any lithography settings, such as aberrations, various illumination schemes and vectorial imaging, as shown in the code examples that come with the package. Because ELIAS can compute TCCs very accurately, as the experiments demonstrate, ELIAS can be used to benchmark other image simulation tools.

The contributions of this chapter are:

- We prove that the discontinuity of illumination and projection functions is the major source of the numerical errors in TCC.
- We introduce the recursive integration method to reduce these errors.
- We further speed up the algorithm by taking advantage of the correlation of the entries in TCC matrices, all without losing accuracy,
- We show in our experiments that the new algorithm can run thousands of times faster than a conventional algorithm to achieve the same level of numerical accuracy and that ELIAS can be used as a benchmark when high numerical accuracy is required.

2.2 TCC Matrix

In order to compute aerial images, TCC is typically computed on a uniform grid in the frequency domain [24, 31]. Let us denote the grid size as $\tilde{\Delta}$. Based on (1.8), we have that

$$\mathcal{K}_0(f, g) = 0, \quad \text{if } f \text{ or } g \text{ is not in } [-1, 1]. \tag{2.1}$$

We can find multiple rectangular regions: inside the regions the illumination function $\mathcal{J}(f, g)$ is not always zero; outside of the regions it is always zero. If the smallest such rectangular region is of the size

$$\sigma_1 \times \sigma_2, \tag{2.2}$$

then, based on (1.5), we know that TCC $\mathcal{T}(f_1, g_1, f_2, g_2)$ is always zero outside a 4-dimensional box of size

$$(2 + 2\sigma_1) \times (2 + 2\sigma_2) \times (2 + 2\sigma_1) \times (2 + 2\sigma_2). \tag{2.3}$$

This means that we need to compute a 4-dimensional TCC matrix, whose entries are

$$\mathcal{T}(i_1\tilde{\Delta}, j_1\tilde{\Delta}, i_2\tilde{\Delta}, j_2\tilde{\Delta}), \tag{2.4}$$

where i_1, j_1, i_2 and j_2 are integers, and $(i_1\tilde{\Delta}, j_1\tilde{\Delta}, i_2\tilde{\Delta}, j_2\tilde{\Delta})$ are in the box (2.3). We will take advantage of the fact that the integrals in a TCC matrix are related to reduce the runtime (see Section 2.4).

2.3 Integration of a Function with Discontinuity

In Section 2.3.1, we demonstrate that the conventional TCC computation method can result in large truncation errors, when there is jump discontinuity in the integrand. We reduce the errors using the recursive integration method in Section 2.3.2.

2.3.1 Truncation Error Analysis

An integral is usually approximated by a finite sum. For example, an integral $I(u)$ over a finite region R

$$I_R(u) = \iint_R u(x, y)\ \mathrm{d}x\mathrm{d}y, \tag{2.5}$$

can be approximated using the *midpoint* numerical integration rule [28] as a summation of the function values on a grid with grid size Δ,

$$I_R(u) \approx M_{R,\Delta}(u) = \Delta^2 \sum_{ij} u(\boxdot_{ij}). \tag{2.6}$$

Here, $\boxdot_{ij}$ denotes a grid point, which is the center of a square as shown in Figure 2.1. The summation is over all the square centers that are in R. $\boxdot_{ij}$ This

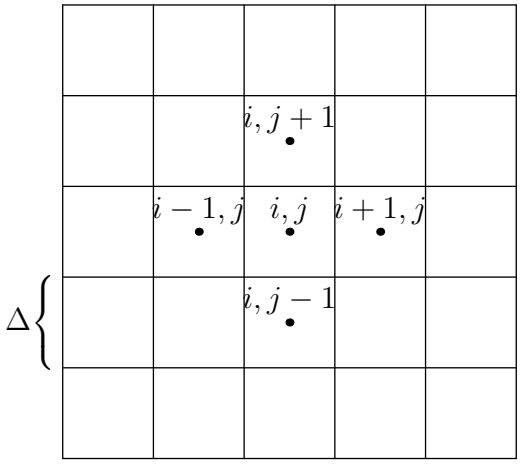


Figure 2.1: Midpoint Rule. Each square, denoted as $\square_{ij}$, is centered at $(i\Delta, j\Delta)$, denoted as $\boxdot_{ij}$.

numerical integration rule has been used previously to compute TCC [54, 92].

In the following theorem, we show that this rule can result in a large truncation error when it is used to integrate a function with jump-discontinuity. The proof is shown in Appendix A.

Theorem 1. *If a function $u(x, y)$ has a bounded support R and is smooth in each connected region of R, and the function and its derivatives to all orders in both arguments are bounded, then the truncation error of $M_{R,\Delta}(u)$ for the approximation of $I_R(u)$ is bounded by*

$$|I_R(u) - M_{R,\Delta}(u)| \leqslant C_1\Delta^2 + C_2\Delta, \tag{2.7}$$

where C_1 and C_2 are two non-negative constants that depend on the function u, but not the grid size Δ.

Here, C_1 is proportional to the area of the support R and is proportional to the average magnitudes of the second order derivatives of the function $u(x, y)$ on R; C_2 is proportional to the length of the boundary of R and is proportional to the average jump of the function $u(x, y)$ on the boundary.

Remark. *When the function $u(x, y)$ is a linear function in each connected region of the support R, the constant C_1 is reduced to zero. In this case, the truncation error is purely bounded by the Δ term, which is originated from the jump of the function $u(x, y)$ along the boundary of the support R. The error is still dominated from the boundary, when the second order derivatives of the function (x, y) are small. Therefore, to improve the accuracy of the numerical integration, we need to give the boundary special treatment, which is discussed in the next subsection.*

2.3.2 Improving Numerical Accuracy — Recursive Integration

We have shown that the numerical integration error is caused mainly by the boundaries. To reduce such errors, we use the recursive integration method. We then estimate the runtime of this method.

We divide the domain of integration into smaller subregions recursively until the approximation in each subregion is accurate enough (Figure 2.3) [71]. Algorithm 1 shows the details. The algorithm concentrates more on the

Algorithm 1 Recursive integration algorithm

```
1: function INTEGRATE(u, □, Δ′)
2:     return Δ² × AVERAGE(u, □, Δ′)

1: function AVERAGE(u, □, Δ′)
2:     Δ ← the size of □
3:     if Δ ⩾ Δ′ and u is not continuous on □ then
4:         Divide the square □ into 4 smaller squares □_i (i = 1, 2, 3, 4). See
   Figure 2.2.
5:         return (1/4) Σ_{i=1}^{4} AVERAGE(u, □_i, Δ′)
6:     else
7:         return u(⊡)
```

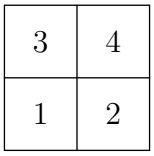


Figure 2.2: The domain of the integration $\square$ can be divided into 4 smaller squares $\square_i$ $(i = 1, 2, 3, 4)$.

boundaries than the internal regions. When the square size is small enough ($< \Delta'$, Δ' is a parameter) or the integrand is continuous in it, the algorithm does not divide the square further. In this case, the algorithm still uses the midpoint rule as an approximation. We denote the approximation of $I_{\square}(u)$ on a boundary square as

$$M_{\square,\Delta'}(u) = \text{INTEGRATE}(u, \square, \Delta'). \tag{2.8}$$

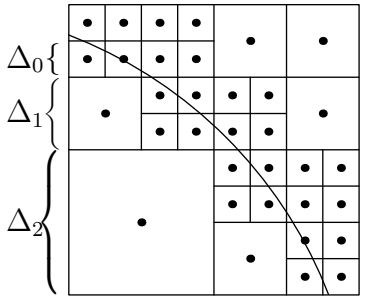


Figure 2.3: The recursive integration method. The integrand is discontinuous on the curve. A square is recursively divided into smaller squares, if the integrand is discontinuous in it. The integrand is evaluated at the not-divided square centers (dots). Δ_k denotes the square size ($k = 0, 1, 2$ in this case).

Therefore, the integral $I_R(u)$ can be approximated as

$$I_R(u) \approx M_{R,\Delta,\Delta'}(u) = \Delta^2 \sum_{ij}^{(0)} u(\boxdot_{ij}) + \sum_{ij}^{(1)} M_{\square_{ij},\Delta'}(u), \tag{2.9}$$

where the first summation is over internal squares and the second is over boundary squares. Compared to the old method (2.6), the new method (2.9) integrates the boundary regions using the recursive integration method instead of the midpoint rule.

The following theorem states the truncation error of this method. The proof is shown in Appendix A.

Theorem 2. *If $u(x, y)$ satisfies all the requirements of $u(x, y)$ that are stated in Theorem 1, then the truncation error of $M_{R,\Delta,\Delta'}(u)$ for the approximation of $I_R(u)$ is bounded by*

$$|I_R(u) - M_{R,\Delta,\Delta'}(u)| \leqslant C_1\Delta^2 + C_2\Delta', \tag{2.10}$$

where C_1 and C_2 here are the same as C_1 and C_2 in (2.7).

Remark. *The only difference between* $M_{R,\Delta}(u)$ *in* (2.6) *and* $M_{R,\Delta,\Delta'}(u)$ *in* (2.9) *is how the integration is done on the boundary squares. The second term in the right hand side of* (2.10) *is also from boundary squares. In the recursive integration method, we can control the minimum square size by* Δ'*. Therefore, that term is related to* Δ' *instead of* Δ *as in* (2.7)*.*

Based on the above theorem, we can reduce the error contributed by boundaries arbitrarily smaller by controlling Δ'. The following theorem shows the runtime of the recursive integration algorithm is related with Δ' according to a power law. In practice, we need to choose an appropriate Δ' to balance the error and the runtime. The proof of the theorem is shown in Appendix A.

Theorem 3. *The time complexity of Algorithm 1 for a square* $\square$ *where* $u(x, y)$ *is discontinuous is*

$$T_{\square,\Delta'} \propto (\Delta/\Delta')^{\gamma},$$

where γ *is a constant satisfying* $0 < \gamma < 2$*.*

Remark. *From the experiments, the constant* γ *can be inferred as shown in Section 2.5.*

2.4 Integration for TCC Matrix

Because we need to compute not just an entry but a whole TCC matrix, we use this property to speed up the algorithm presented in Section 2.3. In Section 2.4.1, we derive that a TCC matrix can be decomposed into a triple correlation term, which is mainly from the internal region, and a correction term, which is from the boundary region. We then show how to compute the two terms efficiently in Section 2.4.2 and 2.4.3.

2.4.1 Numerical Integration Formula

The TCC integral is a continuous triple correlation of the following form

$$h(x_1, y_1, x_2, y_2) = \iint u(x, y)v(x + x_1, y + y_1)w(x + x_2, y + y_2)\,\mathrm{d}x\mathrm{d}y. \quad (2.11)$$

According to the discussion of the TCC matrix in Section 2.2, we need to compute $h(i_1\tilde{\Delta}, j_1\tilde{\Delta}, i_2\tilde{\Delta}, j_2\tilde{\Delta})$ for integers i_1, j_1, i_2, and j_2. We choose $\Delta = \tilde{\Delta}/n$, where n is a positive integer. For any function $u(x, y)$, we denote the function resulted from shifting the arguments of a function $u(x, y)$ as

$$u^{ij}(x, y) = u(x - i\Delta, y - j\Delta).$$

Therefore, we have

$$h(i_1\tilde{\Delta}, j_1\tilde{\Delta}, i_2\tilde{\Delta}, j_2\tilde{\Delta}) = \iint u(x, y)v^{-ni_1,-nj_1}(x, y)w^{-ni_2,-nj_2}(x, y)\,\mathrm{d}x\mathrm{d}y. \quad (2.12)$$

We could use the recursive integration algorithm to compute the approximations of all the TCC matrix entries directly. But we do not for the following reasons:

1. For any function $u(x, y)$,

$$M_{\square_{i1,j1},\Delta'}(u^{i2,j2}) = M_{\square_{i1-i2,j1-j2},\Delta'}(u), \quad (2.13)$$

   which means that shifting the integrand is the same as shifting the region of integration.

2. We need to compute a whole TCC matrix.

The following theorem shows that we can use the fact that the integrand is a product of three functions to reduce the runtime without decreasing accuracy. The proof is shown in Appendix A.

Theorem 4. *If the integrand is a product of a discontinuous function $u(x, y)$ and a continuous function $v(x, y)$ over a square $\square$, we approximate*

$$I_{\square}(uv) = \iint_{\square} u(x, y)v(x, y)\,\mathrm{d}x\mathrm{d}y,$$

by

$$M_{\square,\Delta'}(u, v) = M_{\square,\Delta'}(u)v(\boxdot). \tag{2.14}$$

$I_R(uvw)$ can be approximated as

$$\begin{aligned}
&M_{R,\Delta,\Delta'}(u, v, w)\\
=&\Delta^2 \sum_{ij} \frac{M_{\square_{ij},\Delta'}(u)}{\Delta^2} \frac{M_{\square_{ij},\Delta'}(v)}{\Delta^2} \frac{M_{\square_{ij},\Delta'}(w)}{\Delta^2}\\
&+ \Delta^2 \sum_{ij}^{(2)} \left(\frac{M_{\square_{ij},\Delta'}(vw)}{\Delta^2} - \frac{M_{\square_{ij},\Delta'}(v)}{\Delta^2} \frac{M_{\square_{ij},\Delta'}(w)}{\Delta^2} \right) u(\boxdot_{ij})\\
&+ \Delta^2 \sum_{ij}^{(2)} \left(\frac{M_{\square_{ij},\Delta'}(uw)}{\Delta^2} - \frac{M_{\square_{ij},\Delta'}(u)}{\Delta^2} \frac{M_{\square_{ij},\Delta'}(w)}{\Delta^2} \right) v(\boxdot_{ij})\\
&+ \Delta^2 \sum_{ij}^{(2)} \left(\frac{M_{\square_{ij},\Delta'}(uv)}{\Delta^2} - \frac{M_{\square_{ij},\Delta'}(u)}{\Delta^2} \frac{M_{\square_{ij},\Delta'}(v)}{\Delta^2} \right) w(\boxdot_{ij})\\
&+ \Delta^2 \sum_{ij}^{(3)} \left(\frac{M_{\square_{ij},\Delta'}(uvw)}{\Delta^2} - \frac{M_{\square_{ij},\Delta'}(u)}{\Delta^2} \frac{M_{\square_{ij},\Delta'}(v)}{\Delta^2} \frac{M_{\square_{ij},\Delta'}(w)}{\Delta^2} \right).
\end{aligned} \tag{2.15}$$

The truncation error of $M_{R,\Delta,\Delta'}(u, v, w)$ is of the same order as that of $M_{R,\Delta,\Delta'}(uvw)$.

To simplify the discussions, we introduce a few short hand notations. For a function $u(x,y)$, we define

$$u_{ij} = \begin{cases} \frac{1}{\Delta^2} M_{\square_{ij}}(u), & \text{if } u(x,y) \text{ is continuous in } \square_{ij} \\ 0, & \text{if } u(x,y) \text{ is discontinuous in } \square_{ij} \end{cases},$$

$$\overline{u}_{ij} = \begin{cases} 0, & \text{if } u(x,y) \text{ is continuous in } \square_{ij} \\ \frac{1}{\Delta^2} M_{\square_{ij},\Delta'}(u), & \text{if } u(x,y) \text{ is discontinuous in } \square_{ij} \end{cases},$$

and

$$\hat{u}_{ij} = u_{ij} + \overline{u}_{ij}. \tag{2.16}$$

Note that $\hat{u}$ is a matrix, whereas $\hat{u}_{ij}$, with the index ij, is a number. Based on the above definitions, it is obvious that

$$\overline{u}_{ij} = \overline{u^{-i,-j}}_{00} = \overline{u^{-i,-j}}, \tag{2.17}$$

where we omit the subscript 00 for convenience.

According to Theorem 4, $h_{i_1 j_1 i_2 j_2} = \frac{1}{\Delta^2} h(i_1\tilde{\Delta}, j_1\tilde{\Delta}, i_2\tilde{\Delta}, j_2\tilde{\Delta})$ can be approximated as

$$\begin{aligned} h_{i_1 j_1 i_2 j_2} =& \sum_{ij} \hat{u}_{ij} \hat{v}_{i+ni_1, j+nj_1} \hat{w}_{i+ni_2, j+nj_2} \\ &+ \Big(\sum_{ij}^{(2)} (\overline{v^{-i-ni_1,-j-nj_1} w^{-i-ni_2,-j-nj_2}} - \hat{v}_{i+ni_1,j+nj_1} \hat{w}_{i+ni_2,j+nj_2}) \hat{u}_{ij} \\ &+ \sum_{ij}^{(2)} (\overline{u^{-i,-j} w^{-i-ni_2,-j-nj_2}} - \hat{u}_{ij} \hat{w}_{i+ni_2,j+nj_2}) \hat{v}_{i+ni_1,j+nj_1} \\ &+ \sum_{ij}^{(2)} (\overline{u^{-i,-j} v^{-i-ni_1,-j-nj_1}} - \hat{u}_{ij} \hat{v}_{i+ni_1,j+nj_1}) \hat{w}_{i+ni_2,j+nj_2} \\ &+ \sum_{ij}^{(3)} (\overline{u^{-i,-j} v^{-i-ni_1,-j-nj_1} w^{-i-ni_2,-j-nj_2}} - \hat{u}_{ij} \hat{v}_{i+ni_1,j+nj_1} \hat{w}_{i+ni_2,j+nj_2}) \Big) \\ =& \hat{h}_{i_1 j_1 i_2 j_2} + \tilde{h}_{i_1 j_1 i_2 j_2}. \end{aligned} \tag{2.18}$$

We call the first term, denoted as $\hat{h}_{i_1j_1i_2j_2}$, the triple correlation term, and the sum of the remaining terms, denoted as $\tilde{h}_{i_1j_1i_2j_2}$, the correction term. In the remaining part of this section, we discuss their computation methods.

2.4.2 Triple Correlation Term

$\hat{h}_{i_1j_1i_2j_2}$ in (2.18) can be rewritten as

$$\hat{h}_{i_1j_1i_2j_2} = \sum_{i'=1}^{n}\sum_{j'=1}^{n}\sum_{ij} \hat{u}_{i'+ni,j'+nj}\hat{v}_{i'+n(i+i_1),j'+n(j+j_1)}\hat{w}_{i'+n(i+i_2),j'+n(j+nj_2)} \quad (2.19)$$

where we decompose a summation into a number of summations on grids with a bigger grid size as shown in Figure 2.4 [54]. We can rewrite (2.19) as

$$\hat{h}_{i_1j_1i_2j_2} = \sum_{i'=1}^{n}\sum_{j'=1}^{n}\sum_{ij} {}^{n}\hat{u}_{ij}^{-i',-j'}\,{}^{n}\hat{v}_{i+i_1,j+j_1}^{-i',-j'}\,{}^{n}\hat{w}_{i+i_2,j+j_2}^{-i',-j'}, \quad (2.20)$$

where ${}^{n}\hat{u}_{ij} = \hat{u}_{ni,nj}$, $\hat{u}_{ij}^{i'j'} = \hat{u}_{i-i',j-j'}$ and the matrix ${}^{n}\hat{u}^{ij}$ is the contracted form of ${}^{n}(\hat{u}^{ij})$. We can see that the term

$$\sum_{ij} {}^{n}\hat{u}_{ij}^{-i',-j'}\,{}^{n}\hat{v}_{i+i_1,j+j_1}^{-i',-j'}\,{}^{n}\hat{w}_{i+i_2,j+j_2}^{-i',-j'} \quad (2.21)$$

in (2.20) is a discrete triple correlation.

We show here that the discrete triple correlation can be efficiently computed by the fast Fourier transform (FFT). As a simple case, the continuous 1-dimensional triple correlation can computed by a 2-dimensional convolution [92]

$$\int_{-\infty}^{+\infty} \mathrm{d}f\; u(f)v(f+f')w(f+f'') = (\delta(f_1-f_2)u(-f_1)) * (v(f_1)w(f_2)), \quad (2.22)$$

where $*$ is the convolution operator. Similarly, the discrete 2-dimensional triple correlation can be computed by a 4-dimensional discrete convolution

$$\sum_{ij} \hat{u}_{ij}\hat{v}_{i+i_1,j+j_1}\hat{w}_{i+i_2,j+j_2} = (\delta_{i_1i_2}\delta_{j_1j_2}\hat{u}_{-i_1,-j_1}) * (\hat{v}_{i_1j_1}\hat{w}_{i_2j_2}), \quad (2.23)$$

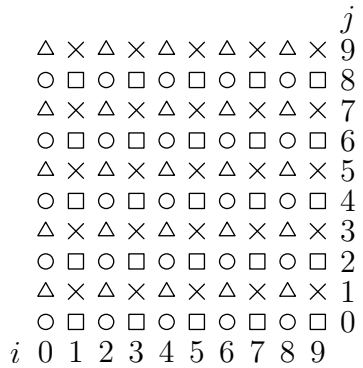


Figure 2.4: As an example ($n = 2$), the summation $\sum_i \sum_j$ can be decomposed into for 4 summations on circles, squares, triangles and crosses. They are $\sum_{\text{even } i} \sum_{\text{even } j}$, $\sum_{\text{odd } i} \sum_{\text{even } j}$, $\sum_{\text{even } i} \sum_{\text{odd } j}$ and $\sum_{\text{odd } i} \sum_{\text{odd } j}$.

where δ_{ij} is the Kronecker delta. The convolution can be computed efficiently by the FFT.

2.4.3 The Correction Term

Using the definition in (2.18), we have a straightforward algorithm to compute the correction term $\tilde{h}_{i_1,j_1,i_2,j_2}$ (Algorithm 2). But this algorithm is slow because of the redundant computation in Line 5, 7 and 9. It can be seen that there can be multiple sets of i, j, i_1 and j_1 such that

$$-i - i_1 = \hat{i}_1$$
$$-j - j_1 = \hat{j}_1$$
$$-i - i_2 = \hat{i}_2$$
$$-j - j_2 = \hat{j}_2$$

for any given $\hat{i}_1, \hat{j}_1, \hat{i}_2$ and $\hat{j}_2$. Therefore, $\overline{v^{-i-i_1,-j-j_1} w^{-i-i_2,-j-j_2}}$ in Line 5 of Algorithm 2 has to be computed multiple times for the same set of superscripts. The same observation is true for $\overline{u^{-i,-j} w^{-i-i_2,-j-j_2}}$ in Line 7 and

Algorithm 2 Straightforward correction term computation algorithm

1: **function** CORRECTION(u, v, w, n)
2: **for all** i_1, j_1, i_2 and j_2 that are multiples of n **do**
3: $\tilde{h}_{i_1/n,j_1/n,i_2/n,j_2/n} \leftarrow 0$
4: **for all** i and j, where $u(x,y)$ is continuous in $\square_{ij}$, $v(x,y)$ is discontinuous in $\square_{i+i_1,j+j_1}$ and $w(x,y)$ is discontinuous in $\square_{i+i_2,j+j_2}$ **do**
5: $\tilde{h}_{i_1/n,j_1/n,i_2/n,j_2/n} \leftarrow (\overline{v^{-i-i_1,-j-j_1} w^{-i-i_2,-j-j_2}} - \hat{v}_{i+i_1,j+j_1}\hat{w}_{i+i_2,j+j_2})\hat{u}_{ij}$
6: **for all** i and j, where $u(x,y)$ is discontinuous in $\square_{ij}$, $v(x,y)$ is continuous in $\square_{i+i_1,j+j_1}$ and $w(x,y)$ is discontinuous in $\square_{i+i_2,j+j_2}$ **do**
7: $\tilde{h}_{i_1/n,j_1/n,i_2/n,j_2/n} \leftarrow (\overline{u^{-i,-j} w^{-i-i_2,-j-j_2}} - \hat{u}_{i,j}\hat{w}_{i+i_2,j+j_2})\hat{v}_{i+i_1,j+j_1}$
8: **for all** i and j, where $u(x,y)$ is discontinuous in $\square_{ij}$, $v(x,y)$ is discontinuous in $\square_{i+i_1,j+j_1}$ and $w(x,y)$ is continuous in $\square_{i+i_2,j+j_2}$ **do**
9: $\tilde{h}_{i_1/n,j_1/n,i_2/n,j_2/n} \leftarrow (\overline{u^{-i,-j} v^{-i-i_1,-j-j_1}} - \hat{u}_{i,j}\hat{v}_{i+i_1,j+j_1})\hat{w}_{i+i_2,j+j_2}$
10: **for all** i and j, where $u(x,y)$ is discontinuous in $\square_{ij}$, $v(x,y)$ is discontinuous in $\square_{i+i_1,j+j_1}$ and $w(x,y)$ is discontinuous in $\square_{i+i_2,j+j_2}$ **do**
11: $\tilde{h}_{i_1/n,j_1/n,i_2/n,j_2/n} \leftarrow \overline{u^{-i,-j} v^{-i-i_1,-j-j_1} w^{-i-i_2,-j-j_2}} - \hat{u}_{i,j}\hat{v}_{i+i_1,j+j_1}\hat{w}_{i+i_2,j+j_2}$

$\overline{u^{-i,-j}v^{-i-i_1,-j-j_1}}$ in Line 9, as well.

In order to reduce the unnecessary computation, we transform the indexes using

$$\begin{cases} i + i_1 \rightarrow i_1 \\ j + j_1 \rightarrow j_1 \\ i + i_2 \rightarrow i_2 \\ j + j_2 \rightarrow j_2 \end{cases} . \tag{2.24}$$

The details are shown in Algorithm 3. Note that the recursive integration is called only once for any set of superscripts in Line 6, 14 and 18 in Algorithm 3. Therefore, the runtime is improved compared with Algorithm 2.

2.5 Results of Experiments

We implement the simulator ELIAS in C++. The simulation platform is a 2.8 GHz Pentium-4 Linux machine. The lithography settings are a normal quadrupole illumination with the parameters $\sigma_{\text{center}} = 0.92$ and $\sigma_{\text{radius}} = 0.15$, and a circular pupil.

We denote the method using the correction term the "new" method, and the method using only the triple correction term the "old" method. We show the accuracy and the runtime of both methods. We demonstrate that the new method is much faster than the old method with the same accuracy requirements.

Algorithm 3 Improved correction term computation algorithm

1: **function** CORRECTION(u, v, w, n)
2: **for all** i_1, j_1, i_2 and j_2 **do**
3: $\tilde{h}_{i_1 j_1 i_2 j_2} \leftarrow 0$
4: **for all** i_1 and j_1, where $v(x, y)$ is discontinuous in $\square_{i_1 j_1}$ **do**
5: **for all** i_2 and j_2, where $i_1 - i_2$ and $j_1 - j_2$ are multiples of n, and $w(x, y)$ is discontinuous in $\square_{i_2 j_2}$ **do**
6: $t \leftarrow \overline{v^{-i_1,-j_1} w^{-i_2,-j_2}} - \hat{v}_{i_1,j_1} \hat{w}_{i_2 j_2}$
7: **for all** i and j, where $i_1 - i$ and $j_1 - j$ are multiples of n **do**
8: **if** $u(x, y)$ is continuous in $\square_{ij}$ **then**
9: $\tilde{h}_{(i_1-i)/n,(j_1-j)/n,(i_2-i)/n,(j_2-j)/n}$ += $t \times \hat{u}_{ij}$
10: **else if** $u(x, y)$ is discontinuous in $\square_{ij}$ **then**
11: $\tilde{h}_{(i_1-i)/n,(j_1-j)/n,(i_2-i)/n,(j_2-j)/n}$ += $\overline{u^{-i,-j} v^{-i_1,-j_1} w^{-i_2,-j_2}} - \hat{u}_{ij} \hat{v}_{i_1 j_1} \hat{w}_{i_2 j_2}$
12: **for all** i and j, where $u(x, y)$ is discontinuous in $\square_{ij}$ **do**
13: **for all** i_2 and j_2, where $i_2 - i$ and $j_2 - j$ are multiples of n, and $w(x, y)$ is discontinuous in $\square_{i_2 j_2}$ **do**
14: $t \leftarrow \overline{u^{-i,-j} w^{-i_2,-j_2}} - \hat{u}_{ij} \hat{w}_{i_2 j_2}$
15: **for all** i_1 and j_1, where $i_1 - i$ and $j_1 - j$ are multiples of n, and $v(x, y)$ is continuous in $\square_{i_1 j_1}$ **do**
16: $\tilde{h}_{(i_1-i)/n,(j_1-j)/n,(i_2-i)/n,(j_2-j)/n}$ += $t \times \hat{v}_{i_1 j_1}$
17: **for all** i_1 and j_1, where $i_1 - i$ and $j_1 - j$ are multiples of n, and $v(x, y)$ is discontinuous in $\square_{i_1 j_1}$ **do**
18: $t \leftarrow \overline{u^{-i,-j} v^{-i_1,-j_1}} - \hat{u}_{ij} \hat{v}_{i_1 j_1}$
19: **for all** i_2 and j_2, where $i_2 - i$ and $j_2 - j$ are multiples of n, and $w(x, y)$ is continuous in $\square_{i_2 j_2}$ **do**
20: $\tilde{h}_{(i_1-i)/n,(j_1-j)/n,(i_2-i)/n,(j_2-j)/n}$ += $t \times \hat{w}_{i_2 j_2}$

2.5.1 Accuracy Verification

We denote the exact solution and the simulation result of $\mathcal{T}(i_1\tilde{\Delta}, j_1\tilde{\Delta}, i_2\tilde{\Delta}, j_2\tilde{\Delta})$ as $\mathcal{T}_{i_1j_1i_2j_2}$ and $\widetilde{\mathcal{T}}_{i_1j_1i_2j_2}$. We denote the error as

$$E_{i_1j_1i_2j_2} = \left|\widetilde{\mathcal{T}}_{i_1j_1i_2j_2} - \mathcal{T}_{i_1j_1i_2j_2}\right|.$$

The worst case (WC) error is defined as

$$E_{\mathrm{WC}} = \max_{i_1j_1i_2j_2} E_{i_1j_1i_2j_2}.$$

The root mean square (RMS) error is defined as

$$E_{\mathrm{RMS}} = \sqrt{\frac{1}{N}\sum_{i_1j_1i_2j_2} E^2_{i_1j_1i_2j_2}}.$$

where N is the number of nonzero $\mathcal{T}_{i_1j_1i_2j_2}$. In the experiments, we took $\tilde{\Delta} = 0.1$.

As we have shown in Theorem 2, the error of TCC is contributed by the internal regions ($C_1\Delta^2$ terms) and the boundaries ($C_2\Delta'$). However, if the integrand is a linear function over the internal regions, the error is only contributed by the boundaries. To analyze both types of errors, we consider an infocus case, where the integrand is constant, and a defocused case ($z = 100\,\mathrm{nm}$), where the integrand is in general not a linear function. In the infocus case, all the errors come from boundaries. In the defocused case, the errors come from both boundaries and internal regions, but we can reduce errors from boundaries by reducing the minimum recursive integration grid size Δ'. From the infocus case, we can determine how small Δ' should be in order to make the errors from boundaries small enough. With a small enough Δ', all the errors practically come from internal regions in the defocused case. By this way, we separate the two types of errors.

For the infocus case, we use the method from [37] to generate the exact solution. It essentially converts TCC region integrals to line integrals, which can be computed analytically. Therefore, it produces results that do not have truncation errors.

Figure 2.5 shows the errors in the new method as functions of Δ'. Obviously, the errors decrease as Δ' decreases and they can be reduced sub-

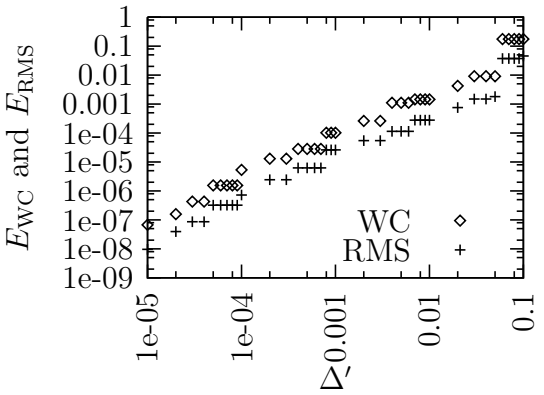


Figure 2.5: Errors for different Δ' of the new method ($n = 1$) (the infocus case).

stantially. Figure 2.6 shows the errors in the old method for different n, the ratio of $\tilde{\Delta}$ to Δ. We can see that the ratio between E_{WC} and E_{RMS} of the old method is a few times greater that the ratio of the new method, which means the TCC matrix errors of the latter case is more evenly distributed than those of the former case. Since $\tilde{\Delta}$ is the same for both methods, when the minimum square size of the old method $\Delta = \frac{\tilde{\Delta}}{n}$ and the minimum square size of the new method Δ' ($\Delta = \tilde{\Delta}$, since $n = 1$ for this case) the same, we should have approximately the same errors. This relation is confirmed by the data replotted in Figure 2.7 (a combination of Figure 2.5 and 2.6).

Because there is no analytical solution available in the literature for

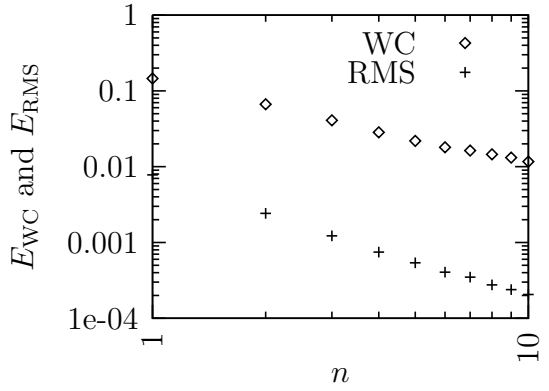


Figure 2.6: Errors for different n of the old method (the infocus case).

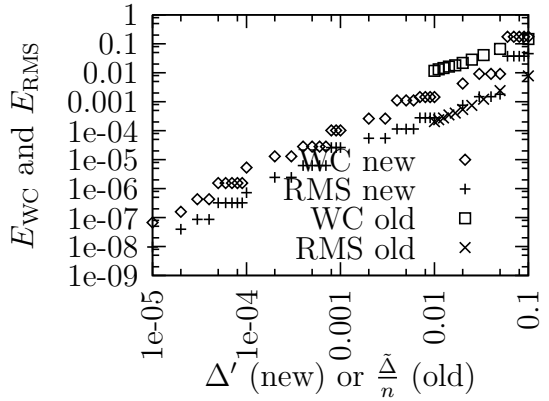


Figure 2.7: Errors of the new (with respect to Δ', $n = 1$) and old (with respect to $\frac{\tilde{\Delta}}{n}$) methods, where $\tilde{\Delta} = 0.1$ (the infocus case).

the defocused case, we chose the results computed with a small enough $\Delta' = 1 \times 10^{-4}$ and a large decimation factor $n = 200$ as the almost exact results. As shown in Figure 2.7, $\Delta' = 1 \times 10^{-4}$ is small enough to bound the errors due to boundaries to the order of about 1×10^{-6}, which is practically very small. In this case, when all the errors from internal regions are much larger than 1×10^{-6}, we can ignore the errors from boundaries as if all the errors are from internal regions. According to Theorem 2 (see the $C_1\Delta^2$ term), the errors shall follow power laws of n. In Figure 2.8, the errors of the old method indeed follow power laws of n, when n is between 1 and 100. The errors of the new method follow power laws of n up to $n \approx 40$. Beyond $n \approx 40$, the errors of the new method are of the order of 1×10^{-6}, in which case the errors from boundaries (when $\Delta' = 1 \times 10^{-4}$) are no longer negligible. It is clear in Figure 2.7 that the errors of the old method are much greater than the errors of the new method for the same n.

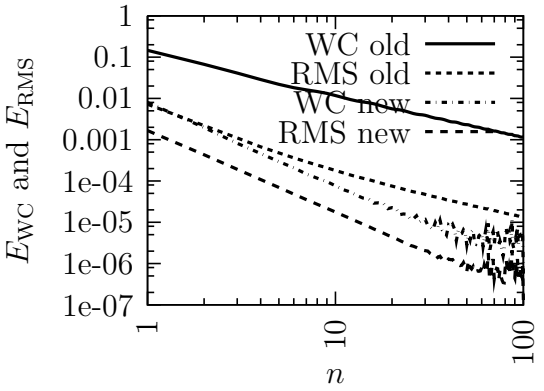


Figure 2.8: Errors of the old and new methods with $\Delta' = 1 \times 10^{-4}$, where $z = 100\,\mathrm{nm}$. Compared with the new method with $n = 200$ and $\Delta' = 1 \times 10^{-4}$ (the defocused case).

2.5.2 Runtime Characteristics

The runtime for the defocusd case shall be the same as that of the infocus case, for the same parameters $\tilde{\Delta}$, n, and Δ', because the same program can be used for both cases. Therefore, we will only show the runtime for the infocus case.

Let us denote the runtime of the computation of the triple correction term using the convolution as $\widehat{T}$, and the runtime of the computation of the correction term using Algorithm 3 as $\widetilde{T}$. The runtimes of both methods are obvious if $\widehat{T}$ and $\widetilde{T}$ are known. Below, we show how the parameters $\tilde{\Delta}$ and n affect the runtime $\widehat{T}$ and how the parameters $\tilde{\Delta}$, n and Δ' affect the runtime $\widetilde{T}$.

Figure 2.9 shows $\widehat{T}$ as a function of n, which demonstrates the relation

$$\widehat{T} \propto n^2. \tag{2.25}$$

This is because the runtime for the discrete correlation (2.21) does not depend

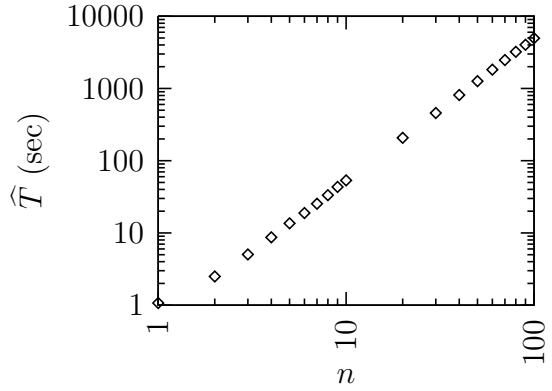


Figure 2.9: $\widehat{T}$ as a function of n ($\tilde{\Delta} = 0.1$). $\widehat{T} \propto n^2$.

on n, but in (2.20) there are n^2 such terms. Figure 2.10 shows $\widehat{T}$ as a function

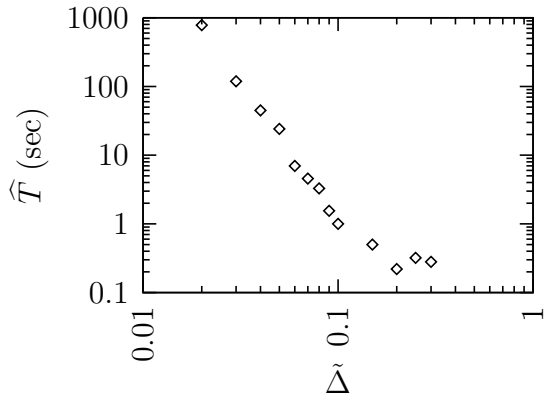


Figure 2.10: $\widehat{T}$ as a function of $\tilde{\Delta}$ ($n = 1$).

of $\tilde{\Delta}$ or equivalently Δ, since $n = 1$. The runtime $\widehat{T}$ is dominated by the FFT used in the convolution, which can be written as

$$\widehat{T} \propto \frac{1}{\Delta^4} \log\Big(\frac{C}{\Delta}\Big)^4,$$

where C is a constant. If the variable Δ is much smaller than C, the change in the log term due to the change in Δ is less important than the term in front of it. Therefore, we can take the log term as a constant, and we have

$$\widehat{T} \propto \frac{1}{\Delta^4},$$

which is consistent with the data in Figure 2.10.

Since the majority of the runtime $\widetilde{T}$ is taken by the recursive integration, $\widetilde{T}$ shall be related to Δ' to a power between 0 and 2, as discussed in Section 2.3.2. Figure 2.11 shows the runtime $\widetilde{T}$ as a function of Δ' ($n = 1$), which can be approximately written as

$$\widetilde{T} \propto \frac{1}{\Delta'}, \tag{2.26}$$

where the power is about 1. Figure 2.12 is a combination of Figure 2.9 and

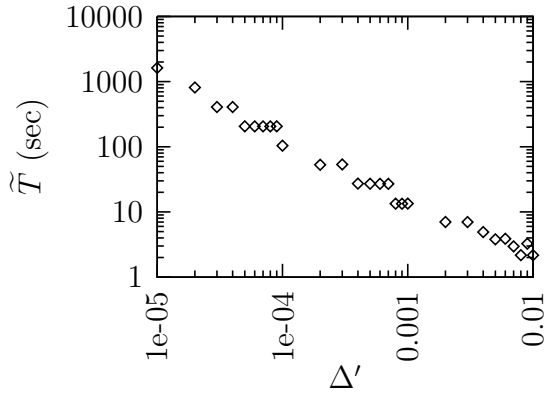


Figure 2.11: $\widetilde{T}$ as a function of Δ' ($\tilde{\Delta} = 0.1$ and $n = 1$).

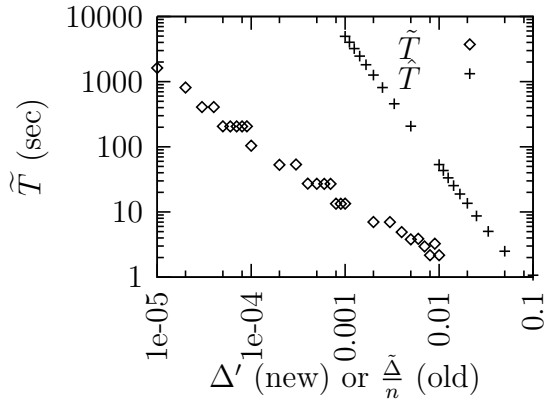


Figure 2.12: Runtimes of the new (with respect to Δ', $n = 1$) and old (with respect to $\frac{\tilde{\Delta}}{n}$) methods, where $\tilde{\Delta} = 0.1$.

Figure 2.11. As we have shown, when Δ' of the new method and $\frac{\tilde{\Delta}}{n}$ of the old method are the same, these two methods generate results of approximately the same accuracy for the infocus case. Figure 2.12 shows that the runtime of the new method can be much faster than that of the old method for the same degree of accuracy. Figure 2.13 shows the runtime $\widetilde{T}$ as a function of $\tilde{\Delta}$ or equivalently Δ, since $n = 1$. The power law relation between $\widetilde{T}$ and

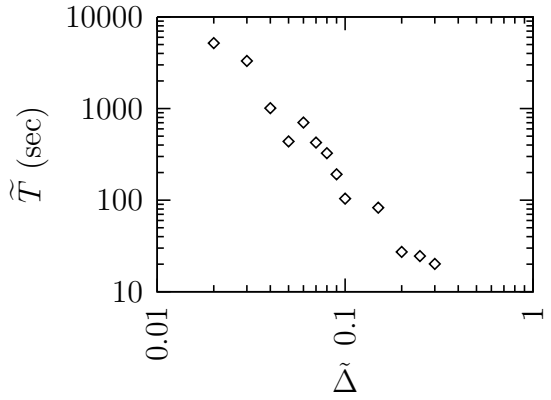


Figure 2.13: $\widetilde{T}$ as a function of $\tilde{\Delta}$ ($n = 1$ and $\Delta' = 1 \times 10^{-4}$).

Δ is due to the fact that the number of the recursive integrations that are computed is proportional to the number of the boundary squares, which is inversely proportional to the square size Δ.

Figure 2.14 shows the runtime for the correction term as a function of n. We can see that there is an optimal n which gives the minimum $\widetilde{T}$. This can be explained in Figure 2.15. When n is small, Δ can be as big as $\tilde{\Delta}$ and there will be some unnecessary recursive integration function calls (represented by gray dots and lines) compared with the case where n is median. When n is big, Δ can be as small as Δ' and the number of the recursive integration function calls reaches the maximum—$(\frac{\tilde{\Delta}}{\Delta'})^2$. Therefore, an optimal runtime is achieved for some median n between 1 and $\frac{\tilde{\Delta}}{\Delta'}$.

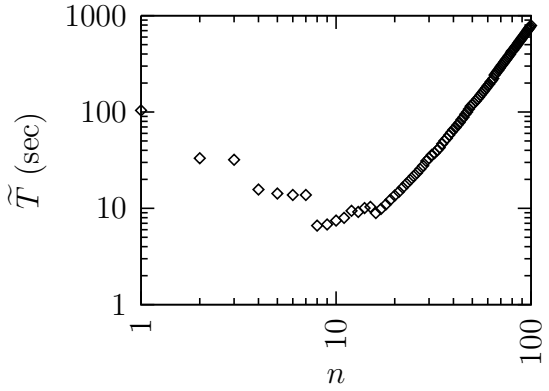
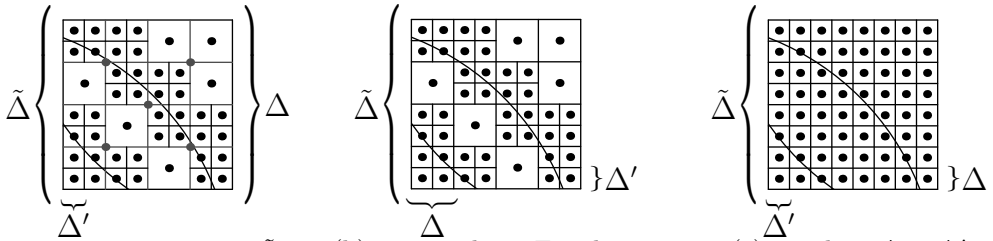


Figure 2.14: $\widetilde{T}$ as a function of n ($\tilde{\Delta} = 0.1$ and $\Delta' = 1 \times 10^{-4}$).



(a) n is small. $\Delta = \tilde{\Delta}$

(b) n is median. For this example, $\Delta = \frac{1}{2}\tilde{\Delta} = 2\Delta'$

(c) n is big. $\Delta = \Delta'$

Figure 2.15: The recursive integration for different n.

Since $\widehat{T}$ increases quadratically with the decrease in the minimum square size (related with n, see (2.25)), while $\widetilde{T}$ increases linearly with the decrease in the minimum square size (related with Δ', see (2.26)), $\widehat{T}$ will be bigger than $\widetilde{T}$ for a small minimum square size. The old method is slower than the new method in this case. For example, the new method with $n = 1$ and $\Delta' = 1 \times 10^{-4}$ give results with the same accuracy as the old method with $n = 1000$ for the infocus case. According to Figure 2.9, we have $\widehat{T}(n = 1) \approx 1\,\text{sec}$, and by extrapolation, we have $\widehat{T}(n = 1000) \approx 3 \times 10^5\,\text{sec}$. According to Figure 2.11, we have $\widetilde{T}(n = 1, \Delta' = 1 \times 10^{-4}) \approx 100\,\text{sec}$. Therefore, the new method speeds up the runtime thousands of times for the infocus case.

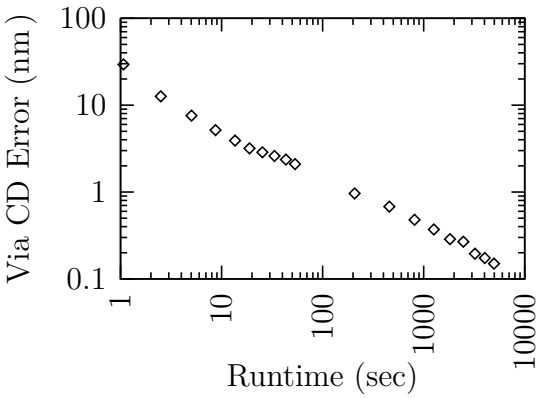
If we choose $\Delta' = 1 \times 10^{-4}$, the error introduced by boundaries in the infocus case can be estimated as 1×10^{-6} (see Figure 2.7), which is also an estimate of the error introduced by boundaries in the defocused case. We require that the total error is bounded to the same order. Therefore, we need to take n about 1000 in the old method, and to take n at least 40 in the new method (see Figure 2.8). According to Figure 2.14, $\widetilde{T}(n = 40, \Delta' = 1 \times 10^{-4}) \approx 40\,\text{sec}$; according to Figure 2.9, $\widehat{T}(n = 40) \approx 1 \times 10^3\,\text{sec}$; and we have estimated $\widehat{T}(n = 1000) \approx 3 \times 10^5\,\text{sec}$. Therefore, the new method speeds up the runtime hundreds of times for the defocused case.

2.5.3 Application to Aerial Image Simulation

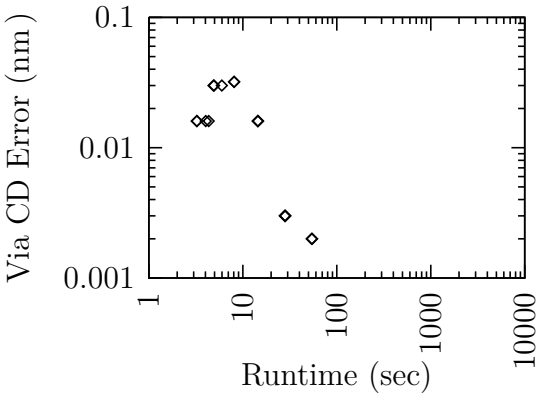
We show below the amount of aerial image errors result from given amounts of TCC computation time. Here, we simulate an isolated via (for a negative resist) of size 105 nm, where the background transmittance is 1 and the feature transmittance is 0. We still use the quadrupole illumination that we mentioned at the beginning of this section. The numerical aperture

NA = 0.8 and the wavelength $\lambda = 193\,\text{nm}$. We choose $\tilde{\Delta} = 0.1$. Critical Dimension (CD) is measured at the threshold of 0.6.

Figure 2.16 and 2.17 show the CD errors as a function of TCC computation runtime using both the old and new methods for the infocus case and the defocused case, respectively. These figures show that the CD errors of



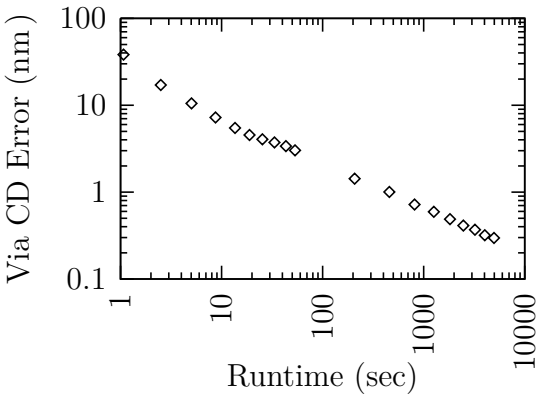
(a) The old method



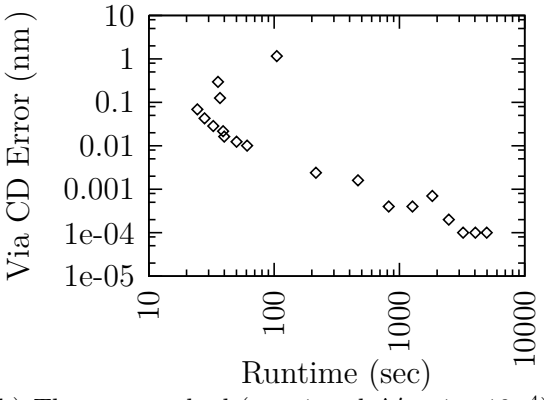
(b) The new method ($n = 1$ and $\Delta' = 1 \times 10^{-4}$). No data points are shown when Runtime is over 100 sec, because the CD errors are almost zero under these conditions.

Figure 2.16: CD errors vs. the TCC computation runtime (the infocus case).

the new method are much less than the errors from the old method with the same amount of TCC computation time. The new method can give almost



(a) The old method



(b) The new method ($n = 1$ and $\Delta' = 1 \times 10^{-4}$)

Figure 2.17: CD errors vs. the TCC computation runtime (the defocused case).

accurate results, for example, 1×10^{-4} nm aerial image CD error, with about an hour TCC computation time, as shown in Figure 2.17. For the same accuracy requirements, the runtime of the old method can be estimated as about a hundred years by extrapolation. Therefore, the new method can be used to benchmark other lithography simulators.

2.6 Summary

We developed an Accurate and Extensible Lithography Aerial Image Simulator (ELIAS). It computes the transmission cross coefficient (TCC) efficiently. The accuracy is increased by using the iterative integration method. The aerial image accuracy is also dramatically improved. ELIAS can be hundreds of times faster than an old method with the same degree of accuracy. Because the results are very accurate (for example, 1×10^{-4} nm aerial image CD error) can be achieved in a reasonable amount of time (for example, hours), ELIAS is ideal to be used to evaluate the numerical errors of other lithography aerial image simulators.

Chapter 3

Fast Image Simulation By Exploiting Lithography System Symmetries

Optimal Coherent Approximations (OCAs) are widely used in full chip simulation, In this chapter, we improve OCAs by considering the symmetric properties of lithography systems. The new method could speed up the runtime by 2× without loss of accuracy. Preliminary results of this work have been published in [70, 106]. We demonstrate that the speedup is applicable to a vectorial imaging model as well. In case the symmetric properties do not hold strictly, the use of the new method can be generalized such that it could still be faster than the old method.

3.1 Introduction

Optimal Coherent Approximations (OCAs) [67, 68] are used for the full-chip image simulation. In this work, we improved this method by exploiting the symmetric properties that are commonly found in lithography systems. The new method could increases the speed of image simulation by 2× without any loss in accuracy. Such improvement can be easily integrated with the other speedup techniques mentioned above.

The main contributions of this chapter are as follows:

- We derive that Hopkins Equation can be reduced such that the imaginary

part of the transmission cross coefficient is not necessary.

- We derive an improved image simulation formula, which could double the speed without lossing any accuracy by using the symmetric properties in common lithography systems.
- The new method works for both scalar and vectorial imaging model.
- The new method still improves runtime even if lithography systems are not perfectly symmetric.

3.2 Symmetries in Lithography Systems

We point out some symmetric properties commonly found in lithography systems, which will be used for the derivation of the new formula.

Eq. (1.10), Figure 1.4 and Eq. (1.7) show that the Guassian blur function $\mathcal{G}(\boldsymbol{k})$, the illumination function $\mathcal{J}(\boldsymbol{k})$ and the projection system transfer function $\mathcal{K}(\boldsymbol{k})$ have the following two properties.

Property 1.

$$\mathcal{G}(\boldsymbol{k}) \in \mathbb{R},\ \mathcal{J}(\boldsymbol{k}) \in \mathbb{R} \text{ and } \mathcal{K}(\boldsymbol{k}) \in \mathbb{C}, \tag{3.1}$$

where $\mathbb{R}$ and $\mathbb{C}$ denote the set of all real numbers and all complex numbers, respectively.

Property 2. *It is reasonable to assume that diffusion is rotational invariant. Then we have*

$$\mathcal{G}(\boldsymbol{k}) = \mathcal{G}(-\boldsymbol{k}). \tag{3.2}$$

Since symmetrical illumination schemes are commonly used (not necessarily limited to the ones in Figure 1.4), we have

$$\mathcal{J}(\boldsymbol{k}) = \mathcal{J}(-\boldsymbol{k}). \tag{3.3}$$

Assuming no odd order aberrations, we have

$$\mathcal{K}(\boldsymbol{k}) = \mathcal{K}(-\boldsymbol{k}). \tag{3.4}$$

Remark. *These properties are true in general and are not necessarily specific only to the forms in Eq.* (1.10)*, Figure 1.4 and Eq.* (1.7)*.*

The mask transmission function $F(\boldsymbol{r})$ is real for commonly used masks, such as binary mask (BIM) or phase shift mask (PSM) with the phases of 0° and 180°. By the definition of the Fourier Transform, we can easily prove the following property of $F(\boldsymbol{r})$'s inverse Fourier Transform $\mathcal{F}(\boldsymbol{k})$.

Property 3.

$$\mathcal{F}(\boldsymbol{k}) = \mathcal{F}^*(-\boldsymbol{k}). \tag{3.5}$$

3.3 The Reduced Hopkins Equation

In this section, we prove a few lemmas on TCC and derive a reduced Hopkins Equation. Note that the proofs are for the case with the diffusion term, but the lemmas are true even if there is no diffusion term.

With Property 1, we have the following lemma.

Lemma 1. $\mathcal{T}(\boldsymbol{k}', \boldsymbol{k}'')$ *is Hermitian, that is,*

$$\mathcal{T}(\boldsymbol{k}', \boldsymbol{k}'') = \mathcal{T}^*(\boldsymbol{k}'', \boldsymbol{k}'). \tag{3.6}$$

Proof.

$$\begin{aligned}
&\mathcal{T}(\boldsymbol{k}', \boldsymbol{k}'') \\
&= \mathcal{G}(\boldsymbol{k}' - \boldsymbol{k}'') \iint \mathcal{J}(\boldsymbol{k}) \mathcal{K}(\boldsymbol{k} + \boldsymbol{k}') \mathcal{K}^*(\boldsymbol{k} + \boldsymbol{k}'') \mathrm{d}^2\boldsymbol{k} \\
&= \Big(\mathcal{G}(\boldsymbol{k}' - \boldsymbol{k}'') \iint \mathcal{J}(\boldsymbol{k}) \mathcal{K}^*(\boldsymbol{k} + \boldsymbol{k}') \mathcal{K}(\boldsymbol{k} + \boldsymbol{k}'') \mathrm{d}^2\boldsymbol{k} \Big)^* \\
&= \mathcal{T}^*(\boldsymbol{k}'', \boldsymbol{k}'). \qquad (3.7)
\end{aligned}$$

□

With Property 2, we have the following lemma.

Lemma 2. *$\mathcal{T}(\boldsymbol{k}', \boldsymbol{k}'')$ is symmetric under the reflection operation about the origin in the frequency domain,*

$$\mathcal{T}(\boldsymbol{k}', \boldsymbol{k}'') = \mathcal{T}(-\boldsymbol{k}', -\boldsymbol{k}''). \qquad (3.8)$$

Proof.

$$\begin{aligned}
&\mathcal{T}(\boldsymbol{k}', \boldsymbol{k}'') \\
&= \mathcal{G}(\boldsymbol{k}' - \boldsymbol{k}'') \iint \mathcal{J}(\boldsymbol{k}) \mathcal{K}(\boldsymbol{k} + \boldsymbol{k}') \mathcal{K}^*(\boldsymbol{k} + \boldsymbol{k}'') \mathrm{d}^2\boldsymbol{k} \\
&= \mathcal{G}\big((-\boldsymbol{k}') - (-\boldsymbol{k}'')\big) \iint \mathcal{J}(-\boldsymbol{k}) \mathcal{K}(-\boldsymbol{k} - \boldsymbol{k}') \mathcal{K}^*(-\boldsymbol{k} - \boldsymbol{k}'') \mathrm{d}^2\boldsymbol{k} \\
&= \mathcal{G}\big((-\boldsymbol{k}') - (-\boldsymbol{k}'')\big) \iint \mathcal{J}(\boldsymbol{k}) \mathcal{K}(\boldsymbol{k} - \boldsymbol{k}') \mathcal{K}^*(\boldsymbol{k} - \boldsymbol{k}'') \mathrm{d}^2\boldsymbol{k} \\
&= \mathcal{T}(-\boldsymbol{k}', -\boldsymbol{k}''). \qquad (3.9)
\end{aligned}$$

□

Remark. *The proofs of Lemma 1 and 2 do not use the particular function forms of $\mathcal{G}$, $\mathcal{J}$ and $\mathcal{K}$ but only Property 1 and 2. These conclusions are generally true for common lithography systems.*

With Lemma 1 and 2, we immediately have the following corollary.

Corollary 1.

$$\mathcal{T}(\boldsymbol{k}', \boldsymbol{k}'') = \mathcal{T}^*(-\boldsymbol{k}'', -\boldsymbol{k}'). \tag{3.10}$$

Using Property 3, we have the following lemma.

Lemma 3. *If* $\mathcal{T}(\boldsymbol{k}', \boldsymbol{k}'') = -\mathcal{T}(-\boldsymbol{k}'', -\boldsymbol{k}')$*, we have* $\mathcal{I}(\boldsymbol{k}) = 0$*.*

Proof. We prove the lemma by proving

$$\mathcal{I}(\boldsymbol{k}) = -\mathcal{I}(\boldsymbol{k}). \tag{3.11}$$

$$\begin{aligned}
\mathcal{I}(\boldsymbol{k}) &= \iint \mathcal{T}(\boldsymbol{k}+\boldsymbol{k}', \boldsymbol{k}')\mathcal{F}(\boldsymbol{k}+\boldsymbol{k}')\mathcal{F}^*(\boldsymbol{k}')\mathrm{d}^2\boldsymbol{k}' \\
&= -\iint \mathcal{T}(-\boldsymbol{k}', -\boldsymbol{k}-\boldsymbol{k}')\mathcal{F}(-\boldsymbol{k}')\mathcal{F}^*(-\boldsymbol{k}-\boldsymbol{k}')\mathrm{d}^2\boldsymbol{k}' \\
&= -\iint \mathcal{T}(\boldsymbol{k}+\boldsymbol{k}', \boldsymbol{k}')\mathcal{F}(\boldsymbol{k}+\boldsymbol{k}')\mathcal{F}^*(\boldsymbol{k}')\mathrm{d}^2\boldsymbol{k}' \\
&= -\mathcal{I}(\boldsymbol{k}).
\end{aligned} \tag{3.12}$$

We replace $-\boldsymbol{k}-\boldsymbol{k}'$ by $\boldsymbol{k}'$ to get the third integral. □

$\mathcal{T}(\boldsymbol{k}', \boldsymbol{k}'')$, as a complex function, can be separated into a real part ($\mathcal{T}_{\text{real}}(\boldsymbol{k}', \boldsymbol{k}'') \in \mathbb{R}$) and an imaginary part ($\mathcal{T}_{\text{imag}}(\boldsymbol{k}', \boldsymbol{k}'') \in \mathbb{R}$),

$$\mathcal{T}(\boldsymbol{k}', \boldsymbol{k}'') = \mathcal{T}_{\text{real}}(\boldsymbol{k}', \boldsymbol{k}'') + i\mathcal{T}_{\text{imag}}(\boldsymbol{k}', \boldsymbol{k}''). \tag{3.13}$$

$\mathcal{I}(\boldsymbol{k})$ can be separated accordingly as

$$\mathcal{I}(\boldsymbol{k}) = \mathcal{I}_{\text{real}}(\boldsymbol{k}) + i\mathcal{I}_{\text{imag}}(\boldsymbol{k}), \tag{3.14}$$

where

$$\mathcal{I}_{\text{real}}(\boldsymbol{k}) = \iint \mathcal{T}_{\text{real}}(\boldsymbol{k}+\boldsymbol{k}', \boldsymbol{k}')\mathcal{F}(\boldsymbol{k}+\boldsymbol{k}')\mathcal{F}^*(\boldsymbol{k}')\mathrm{d}^2\boldsymbol{k}' \tag{3.15}$$

and

$$\mathcal{I}_{\text{imag}}(\boldsymbol{k}) = \iint \mathcal{T}_{\text{imag}}(\boldsymbol{k}+\boldsymbol{k}', \boldsymbol{k}')\mathcal{F}(\boldsymbol{k}+\boldsymbol{k}')\mathcal{F}^*(\boldsymbol{k}')\mathrm{d}^2\boldsymbol{k}'. \tag{3.16}$$

By Corollary 1, we have $\mathcal{T}_{\text{real}}(\boldsymbol{k}', \boldsymbol{k}'')$ and $\mathcal{T}_{\text{imag}}(\boldsymbol{k}', \boldsymbol{k}'')$ are symmetric and antisymmetric, respectively. That is,

$$\mathcal{T}_{\text{real}}(\boldsymbol{k}', \boldsymbol{k}'') = \mathcal{T}_{\text{real}}(-\boldsymbol{k}'', -\boldsymbol{k}') \tag{3.17}$$

and

$$\mathcal{T}_{\text{imag}}(\boldsymbol{k}', \boldsymbol{k}'') = -\mathcal{T}_{\text{imag}}(-\boldsymbol{k}'', -\boldsymbol{k}'). \tag{3.18}$$

Using Lemma 3, we have $\mathcal{I}_{\text{imag}}(\boldsymbol{k}) = 0$. Therefore, we have the following theorem.

Theorem 5. *If $\mathcal{T}(\boldsymbol{k}', \boldsymbol{k}'')$ is symmetric under the reflection operation about the origin in the frequency domain (see Lemma 3.8), the image can be computed by the* Reduced Hopkins Equation*,*

$$\mathcal{I}(\boldsymbol{k}) = \iint \mathcal{T}_{real}(\boldsymbol{k}+\boldsymbol{k}', \boldsymbol{k}')\mathcal{F}(\boldsymbol{k}+\boldsymbol{k}')\mathcal{F}^*(\boldsymbol{k}')\mathrm{d}^2\boldsymbol{k}'. \tag{3.19}$$

Using the above theorem, we can derive the speedup simulation formula in the next section.

3.4 Improved Optimal Coherent Approximations

Optimal Coherent Approximations (OCAs) were derived in [67], which shows that the image can be computed by

$$I(\boldsymbol{r}) = \sum_{n=0}^{\infty} \sigma_n \Big| Q_n **F \Big|^2, \tag{3.20}$$

where F is the mask transmission function, $**$ is the convolution operator, and Q_n's (called kernels) are complex functions. The σ_n's, which are real numbers, are ordered such that

$$|\sigma_0| \geqslant |\sigma_1| \geqslant \cdots \geqslant |\sigma_n| \geqslant \cdots . \tag{3.21}$$

An image can be approximated by using only the first few terms,

$$\widetilde{I}_p(\boldsymbol{r}) = \sum_{n=0}^{p-1} \sigma_n \Big| Q_n **F \Big|^2. \tag{3.22}$$

The error can be estimated as

$$\sup_{\boldsymbol{r}} |I(\boldsymbol{r}) - \widetilde{I}_p(\boldsymbol{r})| \leqslant \sigma_p \|F\|_2^2, \tag{3.23}$$

where $\| \cdot \|_2$ denotes the L^2-norm.

However, Theorem 5 was not used in [67]. When we use Theorem 5, we can prove in Appendix B that the operator $|\cdot|$ is not needed. That is, instead of (3.20), we have

$$I(\boldsymbol{r}) = \sum_{n=0}^{\infty} \sigma'_n (Q'_n **F)^2, \tag{3.24}$$

where F and Q'_n's are all real, and

$$|\sigma'_0| \geqslant |\sigma'_1| \geqslant \cdots \geqslant |\sigma'_n| \geqslant \cdots . \tag{3.25}$$

Note that σ_n and σ'_n, and Q_n and Q'_n may not be the same. As an approximation, we also take the first a few terms for the image simulation

$$\widetilde{I}'_{p'}(\boldsymbol{r}) = \sum_{n=0}^{p'-1} \sigma'_n (Q'_n **F)^2. \tag{3.26}$$

Similarly, the error can be estimated as

$$\sup_{\boldsymbol{r}} |I(\boldsymbol{r}) - \widetilde{I}'_{p'}(\boldsymbol{r})| \leqslant \sigma'_{p'} \|F\|_2^2. \tag{3.27}$$

Based on Eq. (3.23) and Eq. (3.27), we can choose the numbers of terms that are needed (p and p') for a given error requirement ϵ, so that $\sigma_{p+1} < \epsilon$ and $\sigma'_{p'+1} < \epsilon$.

The TCC $\mathcal{T}$ is real when there are no aberrations ($z = 0$ and $\Phi(\boldsymbol{k}) = 0$). In this case, we have

$$Q_n = Q'_n \quad \text{and} \quad \sigma_n = \sigma'_n \tag{3.28}$$

for any n. Therefore, the same numbers of terms ($p = p'$) are needed for the same error requirement ϵ. Since the convolution of a complex function (Q) with a real function (F) is $2\times$ slower than the convolution of two real functions (Q' and F), we get a $2\times$ speedup. The approximated images are always the same

$$\widetilde{I}_p(\boldsymbol{r}) = \widetilde{I}'_{p'}(\boldsymbol{r}).$$

Therefore, there is no loss in the accuracy from (3.22) to (3.26). We have the following corollary.

Corollary 2. *When there are no aberrations,* (3.24) *gives a* $2\times$ *speedup without loss of accuracy.*

Otherwise the speedup is

$$s = \frac{2p}{p'}. \tag{3.29}$$

In Section 3.6, we show experimentally the speedup for some other cases.

3.5 Extensions to Vectorial Imaging and Non-Perfect Symmetries

The above improvement was shown for scalar image modeling with perfect symmetries. We will show below that it also works for vectorial image modeling [100] and non-perfect symmetric lithography systems.

3.5.1 Vectorial Imaging

According to [100], the TCC in the scalar model becomes a TCC matrix in the vectorial model, which can be written as

$$\mathcal{T}_{ij}(\boldsymbol{k}', \boldsymbol{k}'') = \iint \mathcal{J}(\boldsymbol{k})\mathcal{K}(\boldsymbol{k}+\boldsymbol{k}')\mathcal{K}^*(\boldsymbol{k}+\boldsymbol{k}'') \sum_{k=\{\mathrm{x,y,z}\}} \mathcal{M}_{ki}(\boldsymbol{k}+\boldsymbol{k}')\mathcal{M}^*_{kj}(\boldsymbol{k}+\boldsymbol{k}'')\mathrm{d}^2\boldsymbol{k},$$

where

$$\boldsymbol{\mathcal{M}}_0(\boldsymbol{k}) = \boldsymbol{\mathcal{M}}_0(f,g) = \begin{pmatrix} \mathcal{M}_{0\mathrm{xx}} & \mathcal{M}_{0\mathrm{yx}} \\ \mathcal{M}_{0\mathrm{xy}} & \mathcal{M}_{0\mathrm{yy}} \\ \mathcal{M}_{0\mathrm{xz}} & \mathcal{M}_{0\mathrm{yz}} \end{pmatrix} = \begin{pmatrix} \frac{\beta^2+\alpha^2\gamma}{1-\gamma^2} & -\frac{\alpha\beta}{1+\gamma} \\ -\frac{\alpha\beta}{1+\gamma} & \frac{\alpha^2+\beta^2\gamma}{1-\gamma^2} \\ -\alpha & -\beta \end{pmatrix}$$

and

$$\begin{aligned} \alpha &= f\sin\theta_{\mathrm{obj}}, \\ \beta &= g\sin\theta_{\mathrm{obj}}, \\ \gamma &= \sqrt{1-(f^2+g^2)\sin^2\theta_{\mathrm{obj}}}, \end{aligned}$$

where θ_{obj} is the semi-aperture angle at the image plane. For unpolarized illumination, an equivalent TCC can be written as [2, 3]

$$\mathcal{T} = \mathcal{T}_{xx} + \mathcal{T}_{yy}. \tag{3.30}$$

Similar to Lemma 1 and 2, we can check that $\mathcal{T}$ in Eq. (3.30) is Hermitian

$$\mathcal{T}(\boldsymbol{k}', \boldsymbol{k}'') = \mathcal{T}^*(\boldsymbol{k}'', \boldsymbol{k}')$$

and symmetric under the reflection operation about the origin

$$\mathcal{T}(\boldsymbol{k}', \boldsymbol{k}'') = \mathcal{T}(-\boldsymbol{k}', -\boldsymbol{k}'').$$

Therefore, Eq. (3.26) is valid for vectorial imaging as well.

3.5.2 Non-Perfect Symmetries

In practice, lithography systems may not be perfectly symmetric due to some errors. That is, Property 1 and 2 may not hold perfectly. However, we are still able to speed up the simulation. To our best knowledge, it is the first time that this has been demonstrated.

We can separate $\mathcal{T}(\boldsymbol{k}', \boldsymbol{k}'')$ into two parts as

$$\mathcal{T}(\boldsymbol{k}', \boldsymbol{k}'') = \mathcal{T}_{\text{sym}}(\boldsymbol{k}', \boldsymbol{k}'') + \mathcal{T}_{\text{anti}}(\boldsymbol{k}', \boldsymbol{k}''), \tag{3.31}$$

where

$$\mathcal{T}_{\text{sym}}(\boldsymbol{k}', \boldsymbol{k}'') = \frac{\mathcal{T}(\boldsymbol{k}', \boldsymbol{k}'') + \mathcal{T}^*(-\boldsymbol{k}'', -\boldsymbol{k}')}{2} \tag{3.32}$$

and

$$\mathcal{T}_{\text{anti}}(\boldsymbol{k}', \boldsymbol{k}'') = \frac{\mathcal{T}(\boldsymbol{k}', \boldsymbol{k}'') - \mathcal{T}^*(-\boldsymbol{k}'', -\boldsymbol{k}')}{2}. \tag{3.33}$$

It is easy to check that $\mathcal{T}_{\text{sym}}$ is symmetric

$$\mathcal{T}_{\text{sym}}(\boldsymbol{k}', \boldsymbol{k}'') = \mathcal{T}^*_{\text{sym}}(-\boldsymbol{k}'', -\boldsymbol{k}') \tag{3.34}$$

and $\mathcal{T}_{\text{anti}}$ is antisymmetric

$$\mathcal{T}_{\text{anti}}(\boldsymbol{k}', \boldsymbol{k}'') = -\mathcal{T}^*_{\text{anti}}(-\boldsymbol{k}'', -\boldsymbol{k}'). \tag{3.35}$$

Similar to the deduction of Theorem 5, we only need the real part of $\mathcal{T}_{\text{sym}}(\boldsymbol{k}', \boldsymbol{k}'')$ and the imaginary part of $\mathcal{T}_{\text{anti}}(\boldsymbol{k}', \boldsymbol{k}'')$ for the computation of $\mathcal{I}(\boldsymbol{k})$.

Assuming that p terms are needed to decompose $\mathcal{T}$, q_1 terms are needed to decompose $\mathcal{T}_{\text{sym,real}}$, and q_2 terms are needed to decompose $\mathcal{T}_{\text{anti,imag}}$, the runtime speedup is

$$s = \frac{2p}{q_1 + q_2}. \tag{3.36}$$

If lithography systems are close to symmetric, we have that $\mathcal{T}_{\text{sym}}$ is close to $\mathcal{T}$ and $\mathcal{T}_{\text{anti}}$ is small. Therefore, p is close to q_1 and q_2 is much smaller than p. In this case, the speedup is close to $2\times$.

3.6 Results of Experiments

In this section, we numerically validate our previous statements. The implementations were in C++ [105], and simulations were on a 2.8 GHz Pentinum-4 Linux machine.

We used the conventional partially coherent illumination with $\sigma = 0.7$, the numerical aperture NA $= 0.8$, the wavelength $\lambda = 193\,\text{nm}$, and the defocus $z = 100\,\text{nm}$, unless otherwise noted.

3.6.1 Validation of TCC's symmetrical property

For the properties of TCC, we numerically validate only the symmetric property (Lemma 2), because the Hermitian property (Lemma 1) is well known.

We denote $\mathcal{T}(\boldsymbol{k}, \boldsymbol{k}')$ as $\mathcal{T}(f, g, f', g')$, where $(f, g) = \boldsymbol{k}$ and $(f', g') = \boldsymbol{k}'$ to simplify the discussions below. It is easy to check that

$$\mathcal{T}(f, g, f', g') = 0, \quad \text{for} \quad (f, g, f', g') \notin \mathbb{B}, \tag{3.37}$$

where $\mathbb{B}$ is a 4-dimensional box

$$\mathbb{B} = (-1 - \sigma, 1 + \sigma)^4. \tag{3.38}$$

We numerically simulate the TCC on the all points

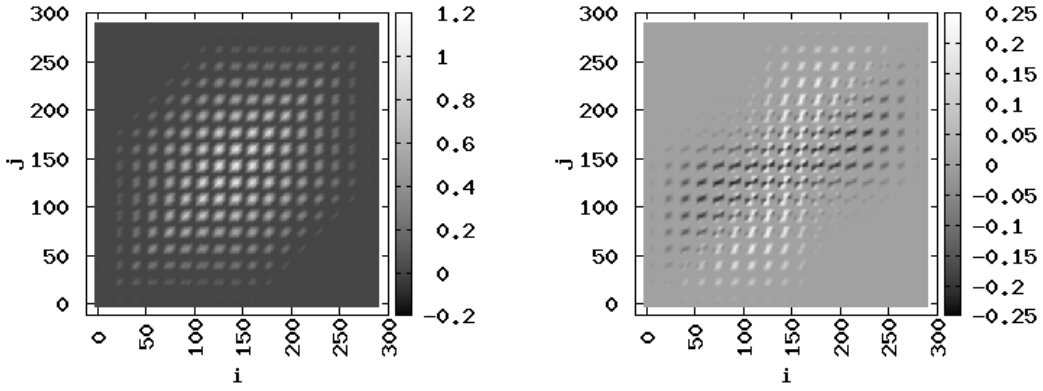
$$(i_1, j_1, i_2, j_2)\Delta \in \mathbb{B}, \tag{3.39}$$

where the grid size in the frequency domain $\Delta = 0.2$, and the numbers i_1, j_1, i_2 and j_2 are integers in the interval $[-N, N]$, where $N = \lfloor \frac{1+\sigma}{\Delta} \rfloor = 8$.

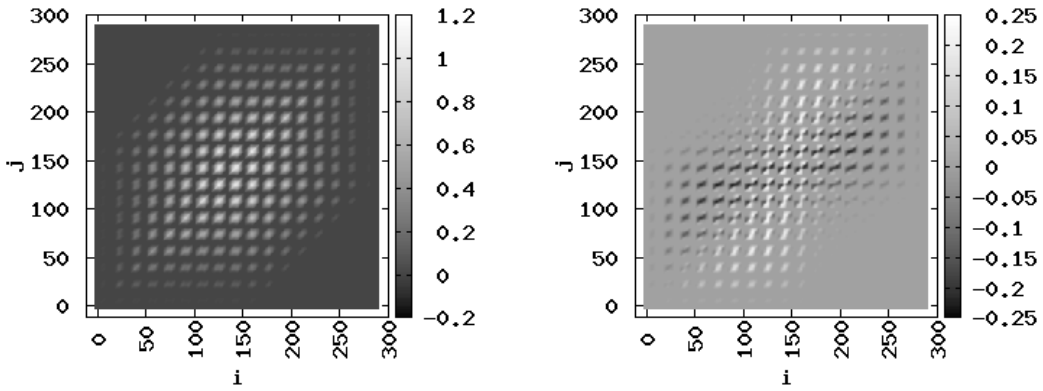
$\mathcal{T}(f, g, f', g')$ is a 4-dimensional function, which needs to be reindexed to draw 2-dimensionally. We denote $\mathcal{T}(i_1\Delta, j_1\Delta, i_2\Delta, j_2\Delta)$ as $\mathcal{T}_{ij}$, where

$$\begin{cases} i = (i_1 + N) + (2N + 1)(i_2 + N), \\ j = (j_1 + N) + (2N + 1)(j_2 + N) \end{cases} \tag{3.40}$$

to help visualize the TCC [24]. The indexes i and j are in $[0, (2N+1)^2 - 1] = [0, 288]$. We also denote $\mathcal{T}(-i_1\Delta, -j_1\Delta, -i_2\Delta, -j_2\Delta)$ as $\widetilde{\mathcal{T}}_{ij}$. We use $\Re$ and $\Im$ to denote the real part and the imaginary part. Figure 3.1 and 3.2 show $\Re(\mathcal{T}_{ij})$, $\Im(\mathcal{T}_{ij})$, $\Re(\widetilde{\mathcal{T}}_{ij})$ and $\Im(\widetilde{\mathcal{T}}_{ij})$ for both the scalar model and the vectorial model. These two figures show that $\widetilde{\mathcal{T}}_{ij} = \widetilde{\mathcal{T}}_{ij}$ for both models. Therefore, Lemma 2 is validated.

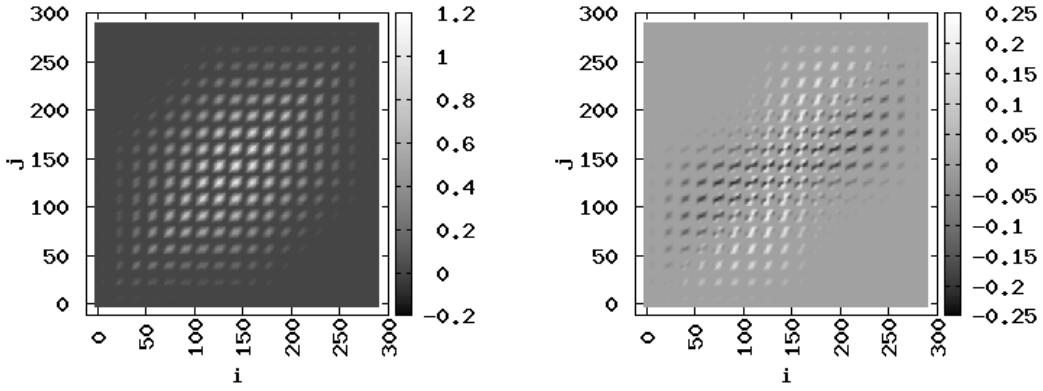


(a) $\Re(\mathcal{T}_{ij})$ (b) $\Im(\mathcal{T}_{ij})$



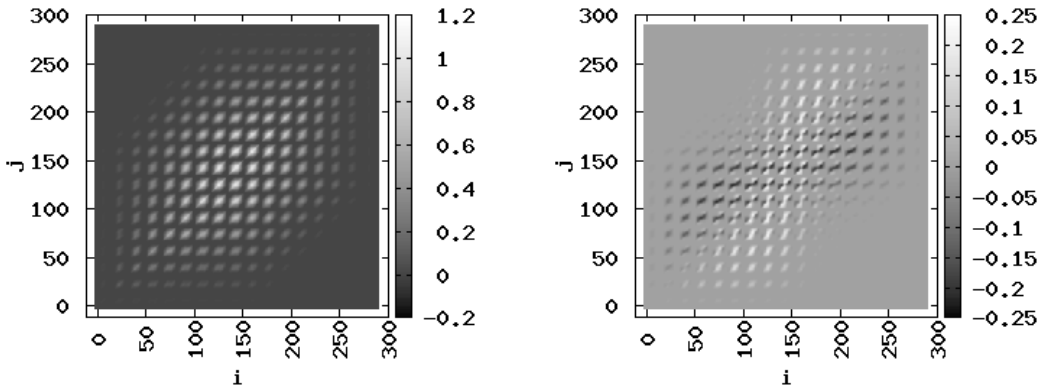
(c) $\Re(\widetilde{\mathcal{T}}_{ij})$ (d) $\Im(\widetilde{\mathcal{T}}_{ij})$

Figure 3.1: Visualization of $\mathcal{T}(\boldsymbol{k}, \boldsymbol{k}')$ and $\mathcal{T}(-\boldsymbol{k}, -\boldsymbol{k}')$ of the *scalar* model ($z = 100\,\mathrm{nm}$). Subfigures (a) and (c) are the same, and Subfigures (b) and (d) are the same. Therefore, $\mathcal{T}(\boldsymbol{k}, \boldsymbol{k}') = \mathcal{T}(-\boldsymbol{k}, -\boldsymbol{k}')$.



(a) $\Re(\mathcal{T}_{ij})$

(b) $\Im(\mathcal{T}_{ij})$



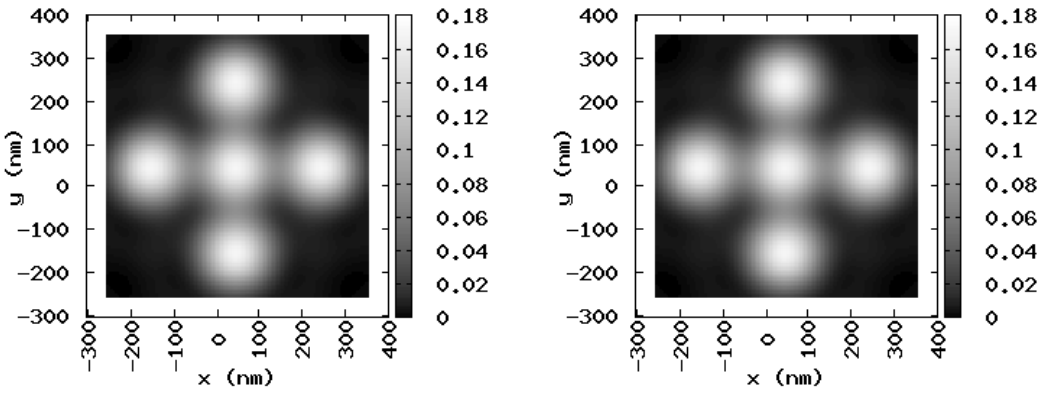
(c) $\Re(\widetilde{\mathcal{T}}_{ij})$

(d) $\Im(\widetilde{\mathcal{T}}_{ij})$

Figure 3.2: Visualization of $\mathcal{T}(\boldsymbol{k}, \boldsymbol{k}')$ and $\mathcal{T}(-\boldsymbol{k}, -\boldsymbol{k}')$ of the *vectorial* model ($z = 100\,\mathrm{nm}$). Subfigure (a) and (c) are the same, and Subfigure (b) and (d) are the same. Therefore, $\mathcal{T}(\boldsymbol{k}, \boldsymbol{k}') = \mathcal{T}(-\boldsymbol{k}, -\boldsymbol{k}')$.

3.6.2 Validation of the Reduced Hopkins Equation

Figure 3.3 shows the simulated image using the scalar model for a five-via pattern using Hopkins Equation and Reduced Hopkins Equation. The



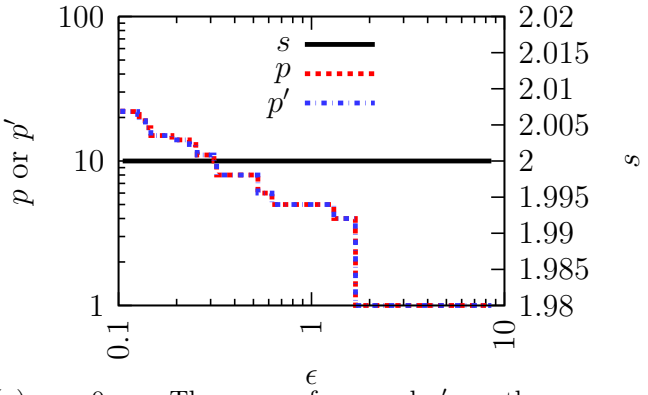
(a) Using Hopkins Equation (b) Using Reduced Hopkins Equation

Figure 3.3: The simulated image for a five-via pattern. Each via is of size 100 nm. The distance between the center via and any other via is 100 nm.

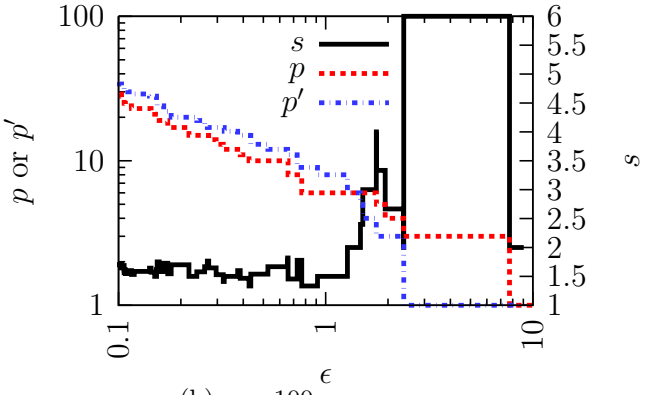
maximum difference between these two images is $1.387\,78 \times 10^{-16}$, which is numerically zero. Therefore, we verified the Reduced Hopkins Equation for the scalar model. The Reduced Hopkins Equation for the vectorial model can also be verified.

3.6.3 Runtime Speedup

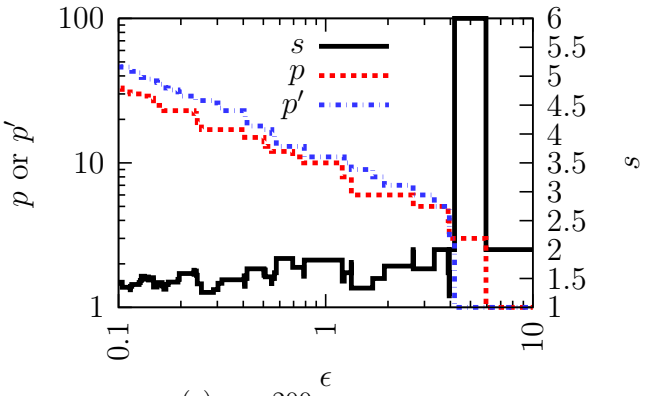
Figure 3.4 shows the numbers of terms p and p', and their ratio as a function of the error requirement ϵ for $z = 100\,\text{nm}$, and $z = 200\,\text{nm}$, respectively (the scalar model). The experiments show that the runtime speedup is $2\times$ for $z = 0\,\text{nm}$. When $z = 100\,\text{nm}$ and $z = 200\,\text{nm}$, the speedup can be



(a) $z = 0\,\text{nm}$. The curves for p and p' are the same.



(b) $z = 100\,\text{nm}$



(c) $z = 200\,\text{nm}$

Figure 3.4: Numbers of terms (p and p') and the runtime speed (using Eq. (3.29)) vs. the error requirement (ϵ).

higher than 2× for some error requirement (ϵ). In the worst case, the speedup is approximately 1.2×.

3.6.4 Non-Perfect Symmetries

When there are odd aberrations, Eq. (3.4) does not hold. Let us consider a small x-coma aberration ($z = 0\,\mathrm{nm}$). The x-coma aberration term $\Phi(\boldsymbol{k})$ is

$$\Phi(\boldsymbol{k}) = \Phi(f, g) = c2\sqrt{2}(3k^2 - 2)f$$

[100], where c is a coefficient (we take it as a small number, 0.01). Figure 3.5 shows the numbers of terms p and p', and their ratio as a function of the error requirement ϵ (the scalar model).

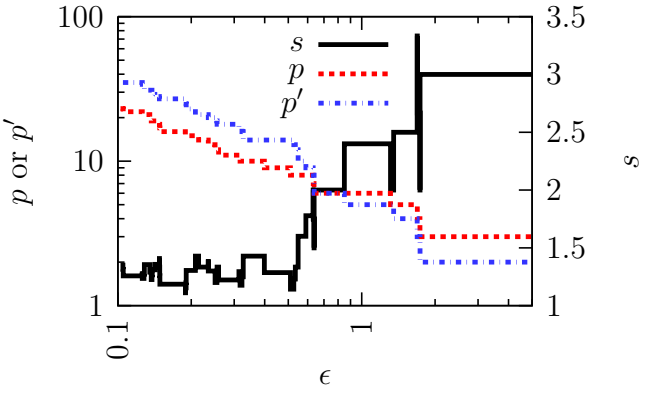


Figure 3.5: Improvement for x-coma with $c = 0.01$.

The speedup s varies as the error requirement ϵ changes, but in the worst case, the speedup is approximately 1.2×, or the speedup can be bigger than 2×.

3.7 Summary

In this chapter, we derive a new method for the lithography simulation, which speeds up OCAs using the symmetric properties of the lithography imaging system. This new method can give 2× speedup if there are no aberrations. It works for both the scalar and the vectorial model. The new method still gives speedup when lithography imaging systems are not perfectly symmetric.

Chapter 4

Variational Lithography Model and Process Variation Aware Optical Proximity Correction

Conventional model-based OPC assumes nominal process conditions without considering process variations because of the lack of variational lithography models. A simple method to improve OPC results under process variations is to sample multiple process conditions across the process window, which requires long runtimes. A variational lithography model (VLIM) is derived, which can simulate across the process window without much runtime overhead compared to the conventional lithography models. A VLIM calibration method is demonstrated to match the model to experimental data. The calibrated model has accuracy comparable to non-variational models, but it has the advantage of taking process variations into consideration. The variational edge placement error (V-EPE) metric is introduced based on the model, a natural extension to the edge placement error (EPE) used in conventional OPC algorithms. A *true* process-variation aware OPC (PV-OPC) framework is proposed using the V-EPE metric. Due to the analytical nature of VLIM, our PV-OPC is only about 2-3× slower than the conventional OPC, but it *explicitly* considers the two main sources of process variations (exposure dose and focus variations) during OPC. Thus our post PV-OPC results are much more robust than the conventional OPC results, in terms of both geometric printability and electrical characterization under process variations.

Preliminary results of this work have been published in [104, 109, 110]. This is a joint work with Dr. Chris A. Mack and Sean X. Shi.

4.1 Introduction

Conventional model-based OPC assumes nominal process parameters [20, 21, 24]. As process variations become more important, OPC software should not disregard them any more. Some primitive attempts have been made in this direction. For example, it is pointed out that the expected contour should be on target [55]. However, no implementation details are provided in this chapter. Defocus aerial images, instead of infocus aerial images, can be used to improve process window robustness [18, 88]. But they rely on extensive lithography simulations to choose the appropriate defocus value, which is very expensive. It is shown in [46] how to modify the OPC algorithm to consider the expected defocus from CMP-induced wafer topography. But again, it is based on a certain defocus condition, without considering focus variations and dose variations. Image-log slope, as an indicator of process sensitivity to dose variations, has been used in [18, 40]. But this approach is incapable of handling focus variations. None of these attempts are aware of the entire process window during OPC. The reason is due to prohibitive runtimes of lithography simulations across the entire process window. Actually, even without considering process variations, it has been reported that model-based OPC software could run for days on multiple computers for a single design [17].

Ignoring OPC impacts or process variations could lead to erroneous timing, power and yield characterization analysis. For example, post-OPC silicon-image-based timing analysis is substantially different from that based

on the drawn layout, e.g., with 36% increase in worst-case slack and significant critical-path reordering [101]. Their analysis is based on OPC with the nominal process. The difference in consideration of process variations probably would be even more [87]. Statistical simulation techniques are demonstrated to map the lithography variability to CD or chip timing [8, 75]. The awareness of across chip line width variations can account for as much as 40% tightening of the best-case to worst-case timing spread [47]. Post-OPC gate non-rectangularity should be considered when estimating timing and leakage more accurately [79]. Therefore, it is important to make the OPC aware of process variations.

In this chapter, we propose a *true process-variation aware OPC* (PV-OPC) framework. Our implementation is based on a sparse OPC algorithm, but the general principle can be applied to dense OPC as well. Our PV-OPC is enabled by the *variational lithography modeling* and guided by the *variational edge placement error* (V-EPE) metric. The main contributions of this chapter are as follows:

- We derive a new analytical variational lithography model, which is generic to handle any focus variation and illumination scheme.
- We provide a variational model calibration method.
- We introduce the concept and obtain the closed-form formulae for the variational EPE metric, and use them to guide our PV-OPC algorithm with explicit consideration of the two main sources of process variations (exposure dose and focus variations).
- The robustness of the PV-OPC algorithm is demonstrated in terms of

both the geometrical and the electrical characterizations compared to conventional OPC.

- The runtime of the PV-OPC algorithm is only about 2-3× that of conventional OPC due to the analytical nature of our models, so it is feasible used it in practice.

4.2 Variations in Lithography System

The term "variation" in lithography systems can refer to the raw process variations, or the derived geometrical and electrical variations, i.e.,

- the distributions of the raw process parameters;
- the severity of change i the print images or circuit parameters (e.g., power and frequency) due to certain amounts of the process parameter changes.

The goal for PV-OPC is, that based on the raw process variations (e.g., dosage and focus), the post PV-OPC image would have good property in terms of derived geometrical or electrical characteristics (e.g., less variations).

There are many manufacturing parameters, e.g., focus error, exposure dose, wavelength (λ), polarization. Great efforts have been made to control lithography system uniformity (over space) and stability (over time). Three kinds of lithography variation sources, dose, focus and mask variations, are believed to be among most important [57]. Chen et al. rigorously related the Mask Error Enhancement Factor (MEEF) to the image log slope [14]. Because image log slope and MEEF indicate CD sensitivities to exposure dose variation, and mask size variation, respectively, mask size variation can be

treated equivalently as exposure dose variation. Thus, we consider only one of these two variations—exposure dose variation. Our experiments show that CD is approximately linearly related to exposure dose variation and quadratically related to focus variation. Assuming higher order terms can be ignored, other first-order parameters can be made equivalent to exposure dose error, and other second-order parameters can be made equivalent to focus error. Therefore, we will focus on the exposure dose and focus variations in this chapter.

4.3 Variational LIthography Model (VLIM)

In Chapter 1, we reviewed the conventional phenomenological lithography models, which require per focus error (z) simulation. To simulate through the range of focus variations, one naive solution is to simulate at many different z values, but this method raises the simulation speed problem and the model consistency issues.

- The simulation time is proportional to the number of discrete z values, which is unaffordable for full chip simulation when the number is large.
- Conventional model calibration methods only work for a single process condition. Models of different z values need separate calibrations. Since measurement errors are unavoidable, two models calibrated at slightly different process conditions might have significant differences in the simulation results.

We address the first problem by introducing a new variational lithography model (VLIM) in Section 4.3.1. The concepts of calibrating the models across the process window have been proposed [78, 90], but no details on the

calibration methods were revealed. To solve the second problem, Section 4.3.2 provides a variational lithography model calibration method in detail. Section 4.4 discusses the fast image simulation for rectilinear polygons using the vertex-based table lookup method.

4.3.1 VLIM Derivation

We have mentioned that exposure dose variation and focus variation are the two most important variations in a lithography system. The exposure dose variation can be transformed to equivalent intensity threshold I_{th} variation, which is easy to be handled. However, the conventional model cannot handle focus variation efficiently. We introduce VLIM to solve this problem. In particular, we derive an analytical formula for the defocus latent image for any illumination schemes by adapting and extending the method used in [93] (which only handles the fully coherent illumination).

The image intensity can be expanded by Taylor Expansion,

$$\mathfrak{I}_G(\boldsymbol{k})\Big|_{z=0} = \sum_{n=0}^{\infty} \frac{z^n}{n!} \frac{\partial^n}{\partial z^n} \mathfrak{I}_G(\boldsymbol{k})\Big|_{z=0} = \sum_{n=0}^{\infty} z^n \mathfrak{I}_{Gn}(\boldsymbol{k}). \tag{4.1}$$

The image intensity sensitivity with respect to the focus error z in a scalar model can be written as [104, 109, 110]

$$\frac{\partial^n}{\partial z^n} \mathfrak{I}(\boldsymbol{k})\Big|_{z=0} = \iint_{-\infty}^{+\infty} \frac{\partial^n}{\partial z^n} \mathfrak{T}(\boldsymbol{k}+\boldsymbol{k}', \boldsymbol{k}')\Big|_{z=0} \mathcal{F}(\boldsymbol{k}+\boldsymbol{k}') \mathcal{F}^*(\boldsymbol{k}') \mathrm{d}^2\boldsymbol{k}'. \tag{4.2}$$

In (4.2), the *variational* TCC $\frac{\partial^n}{\partial z^n} \mathfrak{T}(\boldsymbol{k}', \boldsymbol{k}'')\Big|_{z=0}$ can be computed as

$$\begin{aligned} \frac{\partial^n}{\partial z^n} \mathfrak{T}(\boldsymbol{k}', \boldsymbol{k}'')\Big|_{z=0} =& \mathcal{G}(\boldsymbol{k}' - \boldsymbol{k}'') \sum_{m=0}^{n} (-1)^{n-m} (i2\pi)^n \iint_{-\infty}^{+\infty} \mathcal{J}(\boldsymbol{k}) (\phi(\boldsymbol{k}+\boldsymbol{k}'))^m e^{i2\pi\Phi(\boldsymbol{k})} \mathcal{K}_0(\boldsymbol{k}+\boldsymbol{k}') \\ & \times (\phi(\boldsymbol{k}+\boldsymbol{k}''))^{n-m} e^{-i2\pi\Phi(\boldsymbol{k})} \mathcal{K}_0^*(\boldsymbol{k}+\boldsymbol{k}'') \, \mathrm{d}^2\boldsymbol{k}, \end{aligned} \tag{4.3}$$

where we do not consider other abberations.

Fourier transform both sides of (4.1), we reach the expansion form of the latent image intensity

$$I_G(\boldsymbol{x}) = \sum_{n=0}^{\infty} z^n I_{Gn}(\boldsymbol{x}). \tag{4.4}$$

For binary mask or PSM with phase 0° and 180° (the mask transmission function $F(\boldsymbol{x})$ is always real), when $\Phi(\boldsymbol{k}) = 0$, it can be proved that all the odd terms in (4.4) are equal to zero based on the derivations similar to those in [104]. Then we have

$$I_G(\boldsymbol{x}) = \sum_{n=0}^{\infty} z^{2n} I_{G2n}(\boldsymbol{x}). \tag{4.5}$$

We call the above equation the *defocus latent image expansion*. $I_{Gn}(\boldsymbol{x})$'s are called the variational latent images. It is easy to see I_0 is the infocus ($z = 0$) latent image. (4.5) tells us that the defocus latent image can be expressed as the infocus image plus some correction terms. When z is small and z^n ($n \geqslant 4$) is much smaller than z^2, the higher order terms can be ignored. We get the analytical formula,

$$I_G(\boldsymbol{x}) \cong I_{G0}(\boldsymbol{x}) + z^2 I_{G2}(\boldsymbol{x}). \tag{4.6}$$

We simulated images with the conventional partially coherent illumination ($s = 0.7$), the wavelength $\lambda = 193$nm and the numerical aperture NA $= 0.8$ for a five-bar pattern. Figure 4.1 shows I_G vs. z curves of five randomly chosen locations. We can see that the curves are approximately parabolas in $(-200\,\text{nm}, 200\,\text{nm})$. So (4.6) holds in this range. In general, this range scales as $\frac{\lambda}{\text{NA}^2}$.

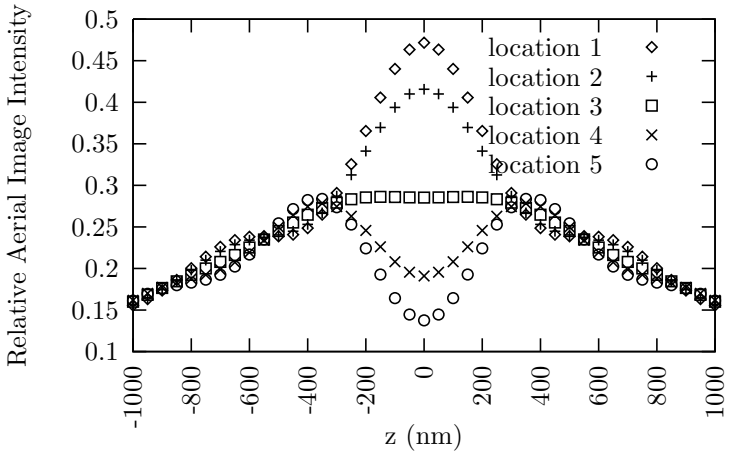
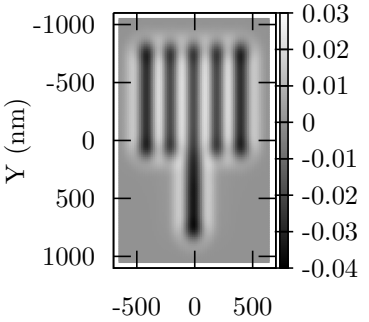


Figure 4.1: Aerial image intensity simulation results (PROLITH$^{\text{TM}}$) at 5 randomly chosen locations.

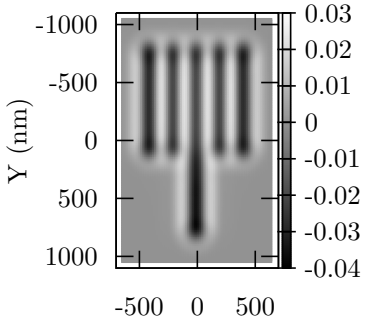
A more rigorous way of deciding the range of z where the approximation (4.6) holds is to compare the magnitude z^2 term and the summation of all higher order terms. Figure 4.2 shows $I_G|_{z=z_0} - I_G|_{z=0}$, $I_{G2}z_0^2$, $I_{G4}z_0^4$, $I_{G2}z_1^2$ and $I_{G4}z_1^4$ of the same five-bar pattern ($z_0 = 100\,\text{nm}, z_1 = 200\,\text{nm}$). $I_{G4}z_0^4$ can be ignored because it is much smaller than $I_{G2}z_0^2$ ($z_0 = 100\,\text{nm}$). For $z_1 = 200\,\text{nm}$, $I_{G2}z_1^2$ is still about 5 times $I_{G4}z_1^4$. Let us say the criterion is $I_{G4}z^4$ can be ignore if it is smaller than one fifth of $I_{G2}z^2$. Then the approximation in (4.6) is appropriate within $\pm 200\,\text{nm}$. Typical lithography simulation shows that this property holds well in a few hundred nm (larger than the typical defocus range in IC manufacturing).

4.3.2 VLIM Calibration

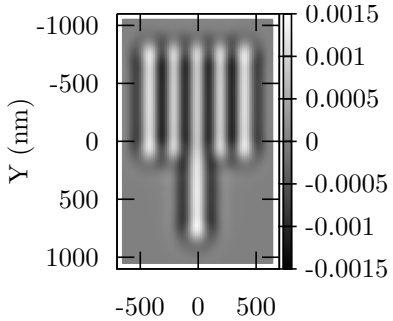
Lithography systems are very complex. For example, PROLITH$^{\text{TM}}$ has pages of physics-based input parameters in its manual [63]. Some parameters



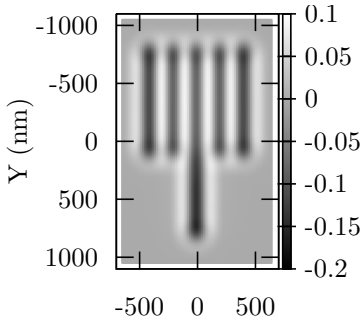
(a) $I_G|_{z=z_0} - I_G|_{z=0}$ image intensity map



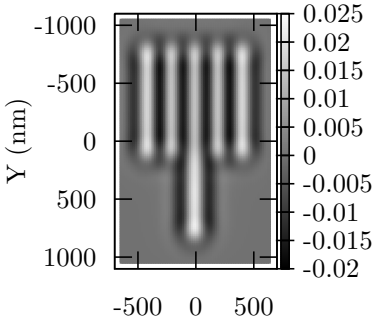
(b) $I_{G2}z_0^2$ image intensity map



(c) $I_{G4}z_0^4$ image intensity map



(d) $I_{G2}z_1^2$ image intensity map



(e) $I_{G4}z_1^4$ image intensity map

Figure 4.2: $I_G|_{z=z_0} - I_G|_{z=0}$ and $I_{G2}z_0^2$ are almost the same. $I_{G2}z_0^2$ is about 20 times $I_{G4}z_0^4$. $I_{G2}z_1^2$ is about 5 times $I_{G4}z_1^4$. ($z_0 = 100\,\text{nm}$ and $z_1 = 200\,\text{nm}$)

in variable threshold resist (VTR) model family are merely fitting parameters and have no physical meaning [41, 76]. In practice, we only care about the consistencies between the model prediction and the experimental data, especially for OPC softwares. Therefore, instead of measuring each individual parameter, the model parameters are usually fit to match the experiment. In this section, we show a VLIM calibration method.

We denote the experimentally measured exposure dose and focus error as E and Z. We fit VLIM to the experimental data using nonlinear regression method. Therefore, we can determine the four input parameters of VLIM, the intensity threshold I_{th}, the focus error z, the diffusion length d and the constant edge bias B.

Assuming the photoresist has a threshold behavior, the intensity threshold I_{th} is inversely proportional to exposure dose E [10]. Taking into account the offset in the exposure dose measurement,

$$I_{\text{th}} = \frac{\beta_E}{E - \alpha_E} \tag{4.7}$$

describes the relationship between E and I_{th} [32, 33, 103]. Because the refraction index of the photoresist film is not 1, the focus error should be scaled by a factor β_Z. α_Z represents the focus measurement offset. So we have

$$z = \beta_Z(Z - \alpha_Z). \tag{4.8}$$

Those relations are used in the model fitting method.

Since the CD responds differently to process variations for different mask pattern $\boldsymbol{P}$, many patterns should be measured in the experiment. In order to predict the CD across process windows, we need to take measurements at various process conditions as well. Suppose the $\text{CD}_{\boldsymbol{P},i}$ is the CD measure

value at exposure dose E_i and focus error Z_i for mask pattern $\boldsymbol{P}$. We can estimate the parameters d, B, α_E, β_E, α_Z and β_Z by minimizing

$$\chi^2(d, B, \alpha_E, \beta_E, \alpha_Z, \beta_Z) \equiv \sum_{\boldsymbol{P}} \sum_i \left(\mathrm{CD}^{\boldsymbol{P}}_{\mathrm{VLIM}}\Big(\frac{\beta_E}{E_i - \alpha_E}, \beta_Z(Z_i - \alpha_Z), d \Big) - 2B - \mathrm{CD}_{\boldsymbol{P},i} \right)^2, \tag{4.9}$$

where $\mathrm{CD}^{\boldsymbol{P}}_{\mathrm{VLIM}}(I_{\mathrm{th}}, z, d)$ is CD function based on VLIM for the mask pattern $\boldsymbol{P}$.

To solve the minimization problem in (4.9), we require that $\mathrm{CD}^{\boldsymbol{P}}_{\mathrm{VLIM}}$'s be analytical functions. Appendix C shows how to generate the analytical functions based on VLIM simulation results.

We estimate the fitting error by computing the standard deviation

$$\sigma = \sqrt{\frac{\chi^2}{N - M}}, \tag{4.10}$$

where N is the number of data points and M is the number of fitting parameters (6 in this case). An example of VLIM calibration is shown Section 4.7.

4.4 Vertex Based Table-Lookup

In this section, we show how to compute I_{G0} and I_{G2} by the table-lookup method. A few tables can be computed from the kernels (defined in Section 1.3) by a method that is similar to [64].

We propose a vertex based convolution method instead of the rectangle based method used in [64] because it requires fewer table-lookup times as shown in the example below. Algorithm 4 shows how to compute the variational latent images from the lookup-tables. The region where a kernel $Q(\boldsymbol{x})$ is non-zero is called the support region whose size is a few times of $\frac{\lambda(1+s)}{\mathrm{NA}}$.

Algorithm 4 Vertex based table-lookup

1: **function** VERTEX BASED TABLE-LOOKUP(The lookup-tables generated from kernels, the decomposed mask in the form of polygons)
2: Retrieve the polygons which intersect the $\mathfrak{O}$'s support region $\mathfrak{R}$
3: Compute the vertices of the polygons with non-zero convolution value
4: Compute the sign of the convolution of each vertex
5: Look up the tables to get the convolution values
6: Compute the variational aerial images I_{G0} and I_{G2} at point $\mathfrak{O}$ by summing up these values

To compute the convolution $Q{*}{*}F$ at point $\mathfrak{O}$, we need only the mask shapes overlapping Q's support region $\mathfrak{R}$, which in turn can be expressed as the summation of the convolution values of each mask shape (Figure 4.3).

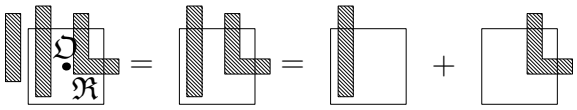


Figure 4.3: Mask truncation and decomposition.

For rectilinear polygons, we can compute the convolutions based on the vertices. As shown in Figure 4.4, any rectilinear polygon convolution can be decomposed into the summation of the convolutions of the regions to the upper-right of each vertex. We store the convolutions of all the upper-right rectangles within the support region in a lookup-table for each kernel.

For the example in Figure 4.5, the contributions of B, C, E and F are zero. Only A's and D's convolutions are needed. If the method in [64] is used, four table-lookups will be needed. It is clear that the vertex-based convolution method is much better.

Both I_{G0} and I_{G2} can be computed by this method. Therefore, the runtime of computing variational latent images (I_{G0} and I_{G2}) will be in the

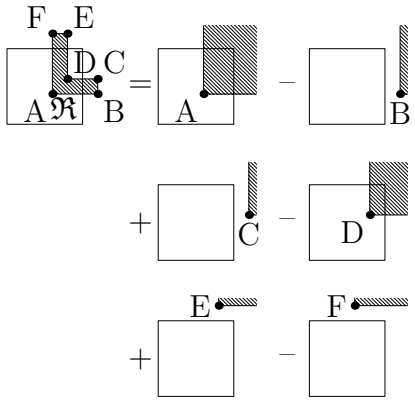


Figure 4.4: Vertex based rectilinear polygon convolution.

same order as the runtime for the non-variational latent image.

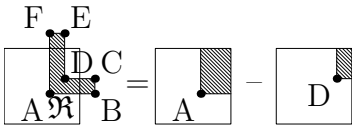


Figure 4.5: Lookup tables store the convolutions of all the upper-right rectangles within the support region. Convolutions with zero-contribution will not be stored.

4.5 Variational EPE (V-EPE) Metrics

For any given point on the target contour, we define its Edge Placement Error (EPE) as the displacement between that point and its nearest printed contour point (Figure 4.6). Note that the EPE defined here is a vector, which is slightly different from the conventional scalar EPE definition in [24] and our previous work [109]. We denoted the EPE for any point A on the target as $\boldsymbol{E}_A$.

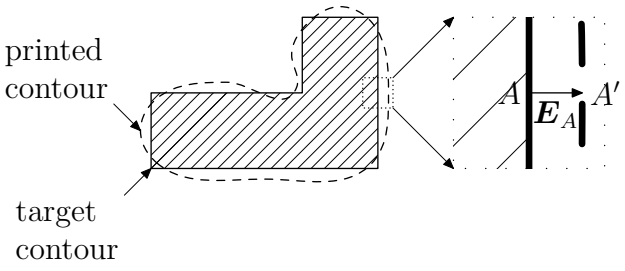


Figure 4.6: EPE concept. The target and printed contours are solid and dashed respectively. The right subfigure is a zoom-in subregion of the left one. A is a point on the target contour. A' is A's closest point on the printed contour. The EPE for A is shown as $\boldsymbol{E}_A = \overrightarrow{AA'}$.

The printed contour is uniquely determined if EPEs of all the target contour points are given.

Conventionally, people do not consider how EPE varies resulting from process variations. In this section, we derive the analytical variational EPE metric (V-EPE) based on VLIM to describe this variation. Our process variation aware OPC will be based on the V-EPE metric, while the conventional OPC is based on the nominal EPE.

4.5.1 Variational EPE Model

We set the bias $B = 0$ to simplify the V-EPE Model derivation, because it is easy to adjust $\boldsymbol{E}_A$ if $B \neq 0$. We drop the subscript $_A$ in $\boldsymbol{E}_A$ and the subscript $_G$ in I_G to simplify the notation in the following derivations. Since there is a one-to-one mapping between the intensity threshold and the exposure dose [32] (see (4.7) for an example), we substitute the intensity threshold I_{th} for the exposure dose.

At a certain intensity threshold, the printed contour is the least sensitive

to the focus variations. This intensity threshold is called the *iso-focal threshold* denoted as $I_{\text{th}_{\text{iso}}}$. The formal definition is

$$\frac{\partial \boldsymbol{E}|_{I=I_{\text{th}_{\text{iso}}}}}{\partial z} = 0, \tag{4.11}$$

which means

$$\left.\frac{\partial I}{\partial z}\right|_{I=I_{\text{th}_{\text{iso}}}} = 0. \tag{4.12}$$

Since $I = I_0 + z^2 I_2$ (see (4.6)), we have

$$I_2|_{I_0=I_{\text{th}_{\text{iso}}}} = 0. \tag{4.13}$$

If we choose $I_{\text{th}_{\text{iso}}}$ as the intensity threshold I_{th}, we have

$$\boldsymbol{E}(I_{\text{th}_{\text{iso}}}, 0) = \boldsymbol{E}(I_{\text{th}_{\text{iso}}}, z). \tag{4.14}$$

We call the above quantity the *iso-focal EPE* and denote it as $\boldsymbol{E}_{\text{iso}}$. Negate both sides of (4.14) and add $\boldsymbol{E}(I_{\text{th}}, z)$ to both sides, and we have

$$\boldsymbol{E}(I_{\text{th}}, z) - \boldsymbol{E}(I_{\text{th}_{\text{iso}}}, 0) = \boldsymbol{E}(I_{\text{th}}, z) - \boldsymbol{E}(I_{\text{th}_{\text{iso}}}, z). \tag{4.15}$$

Approximating $\boldsymbol{E}(I_{\text{th}}, z) - \boldsymbol{E}(I_{\text{th}_{\text{iso}}}, z)$ as a separable function [32], we have

$$\boldsymbol{E}(I_{\text{th}}, z) - \boldsymbol{E}_{\text{iso}} = \boldsymbol{a}(z) b(I_{\text{th}} - I_{\text{th}_{\text{iso}}}), \tag{4.16}$$

where $b(\cdot)$ satisfies $b(0) = 0$.

For small I_{th} variations (usually within 10% for modern lithography systems), $b(I_{\text{th}} - I_{\text{th}_{\text{iso}}})$ can be linearized. So we have

$$\boldsymbol{E}(I_{\text{th}}, z) - \boldsymbol{E}_{\text{iso}} = \boldsymbol{a}(z)(I_{\text{th}} - I_{\text{th}_{\text{iso}}}). \tag{4.17}$$

Due to the $z \leftrightarrow -z$ symmetry of VLIM, $\boldsymbol{a}(z)$ can be expanded for small z's as

$$\boldsymbol{a}(z) = \boldsymbol{a}_0 + \boldsymbol{a}_1 z^2. \tag{4.18}$$

We call the EPE as $\boldsymbol{E}_{\mathrm{t}}$ if the printed contour coincides the target contour. Plugging $\boldsymbol{E}_{\mathrm{t}}$ in (4.17) and using (4.6) and (4.18), we have

$$\begin{aligned} \boldsymbol{E}_{\mathrm{t}} - \boldsymbol{E}_{\mathrm{iso}} = (\boldsymbol{a}_0 + \boldsymbol{a}_1 z^2) \\ \times \big(I_0(\boldsymbol{E}_{\mathrm{t}}) + z^2 I_2(\boldsymbol{E}_{\mathrm{t}}) - I_{\mathrm{th}_{\mathrm{iso}}}\big). \end{aligned} \tag{4.19}$$

Expanding (4.19) with respect to z, we have

$$\begin{aligned} \boldsymbol{E}_{\mathrm{t}} - \boldsymbol{E}_{\mathrm{iso}} = \boldsymbol{a}_0\big(I_0(\boldsymbol{E}_{\mathrm{t}}) - I_{\mathrm{th}_{\mathrm{iso}}}\big) \\ + \Big(\boldsymbol{a}_1\big(I_0(\boldsymbol{E}_{\mathrm{t}}) - I_{\mathrm{th}_{\mathrm{iso}}}\big) + \boldsymbol{a}_0 I_2(\boldsymbol{E}_{\mathrm{t}})\Big) z^2 \\ + O(z^4). \end{aligned} \tag{4.20}$$

We ignore the highest order term of z (the z^4 term). Since the equality in (4.20) is independent of z, by setting the coefficients of z^0 and z^2 to zeros, we get the solutions for $\boldsymbol{a}_0$ and $\boldsymbol{a}_1$

$$\begin{cases} \boldsymbol{a}_0 &= \frac{\boldsymbol{E}_{\mathrm{t}} - \boldsymbol{E}_{\mathrm{iso}}}{I_0(\boldsymbol{E}_{\mathrm{t}}) - I_{\mathrm{th}_{\mathrm{iso}}}} \\ \boldsymbol{a}_1 &= -\boldsymbol{a}_0 \frac{I_2(\boldsymbol{E}_{\mathrm{t}})}{I_0(\boldsymbol{E}_{\mathrm{t}}) - I_{\mathrm{th}_{\mathrm{iso}}}} \end{cases}. \tag{4.21}$$

The vector $\boldsymbol{a}_1$ is propotional to the vector $\boldsymbol{a}_0$ as shown in the second equation of (4.21). We express the ratio between them as $a_1 = -\frac{I_2(\boldsymbol{E}_{\mathrm{t}})}{I_0(\boldsymbol{E}_{\mathrm{t}}) - I_{\mathrm{th}_{\mathrm{iso}}}}$. Then, variational EPE model under any intensity threshold and focus variation (4.17) can be written as

$$\boldsymbol{E}(I_{\mathrm{th}}, z) = \boldsymbol{E}_{\mathrm{iso}} + \boldsymbol{a}_0(1 + a_1 z^2)(I_{\mathrm{th}} - I_{\mathrm{th}_{\mathrm{iso}}}), \tag{4.22}$$

where

$$\begin{cases} \boldsymbol{a}_0 &= \frac{\boldsymbol{E}_{\mathrm{t}} - \boldsymbol{E}_{\mathrm{iso}}}{I_0(\boldsymbol{E}_{\mathrm{t}}) - I_{\mathrm{th}_{\mathrm{iso}}}} \\ a_1 &= -\frac{I_2(\boldsymbol{E}_{\mathrm{t}})}{I_0(\boldsymbol{E}_{\mathrm{t}}) - I_{\mathrm{th}_{\mathrm{iso}}}} \end{cases}. \tag{4.23}$$

4.5.2 Variational-EPE Metrics

From the variational EPE model (4.22), we can compute the V-EPE metric of interest to guide OPC. As an example, let us assume z and I_{th} are independent and normally distributed:

$$z \sim N(\mu_z, \sigma_z^2) \qquad \text{and} \qquad I_{\text{th}} \sim N(\mu_{I_{\text{th}}}, \sigma_{\text{th}}^2), \tag{4.24}$$

we can compute all the EPE moments easily. Assuming $\mu_z = 0$, from (4.22) and (4.24), we have the average EPE (the first moment) under the intensity threshold and focus variations

$$\begin{aligned} \text{V-EPE} = \langle \boldsymbol{E} \rangle &= \boldsymbol{E}_{\text{iso}} + \boldsymbol{a_0}(1 + a_1\sigma_z^2)(\mu_{I_{\text{th}}} - I_{\text{th}_{\text{iso}}}) \\ &= \underbrace{\boldsymbol{E}_{\text{iso}} + \boldsymbol{a_0}(\mu_{I_{\text{th}}} - I_{\text{th}_{\text{iso}}})}_{\boldsymbol{E}_{\text{nom}}} + \underbrace{\boldsymbol{a_0} a_1 \sigma_z^2 (\mu_{I_{\text{th}}} - I_{\text{th}_{\text{iso}}})}_{\boldsymbol{\mu}}, \end{aligned} \tag{4.25}$$

where $\boldsymbol{E}_{\text{nom}}$ is the nominal EPE. It is clear that considering focus variation the average EPE $\langle \boldsymbol{E} \rangle$ will always be different from the nominal EPE $\boldsymbol{E}_{\text{nom}}$. Note that the definition of V-EPE is not limited to the average EPE. Other desirable quantities, such as the variance, can also be included.

For the real manufacturing process, as long as the joint distribution of measured exposure dose (E) and focus error (Z) is available, the joint distribution of I_{th} and z can be computed using (4.7) and (4.8). Therefore, the average EPE $\langle \boldsymbol{E} \rangle$ can be computed without any difficulty.

4.6 Process Variation-aware OPC Algorithm (PV-OPC)

Conventional OPC softwares try to reduce the nominal EPE. However, this would result in more average post-OPC EPE under process variations. Instead, our process variation aware OPC (PV-OPC) algorithm is based on

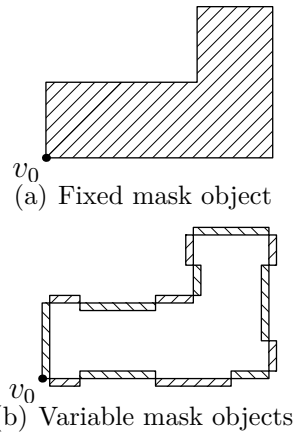
the V-EPE metric defined in the previous section to make the average on target. The metric is generic enough to apply to any sparse OPC algorithm. We show our implementation details in this section.

4.6.1 OPC Shape Engine

"OPC shape engine" refers to the representation, storage and lithography simulator interpretation of mask shapes. Although there are many OPC papers, most of them focused on OPC recipes for commercial OPC softwares, such as Calibre^{TM} or Proteus^{TM}. Only a few early papers [20, 21, 24] discussed the OPC shape engine data structure. In these papers, the original drawn shapes are represented as polygons, called *fixed mask object*s. Many so called *variable mask object*s are attached to the edges of each polygon (Figure 4.7). It can be seen that, to the first order, the simulation time is proportional to the number of vertices (Section 4.4). However, the vertices (e.g., v_0) in this method essentially present multiple times in both the fixed mask object and its variable mask objects, which results in inefficiency during computation.

To get rid of the redundancies in the representation, we employ a similar idea to Chain Code [13, 53, 61]. Our proposed method parametrizes the polygon such that it can efficiently represent changes in the edge locations. We only discuss rectilinear polygons (Figure 4.8 and Figure 4.9). The method can be easily extended to polygons with 45° degree edges.

The rectilinear polygon in Figure 4.8 is composed of a series of directed edges. The head of each edge e_i is connected to the tail of the next one ($e_{\ \mathrm{mod}\ (i+1,N)}$), where N is the number of edges and " mod " denotes the modulo operation. Each edge can be specified by its length l and two Boolean variables h and p, where h indicates whether it is horizontal or vertical, and



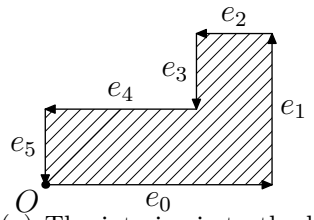
(a) Fixed mask object

(b) Variable mask objects

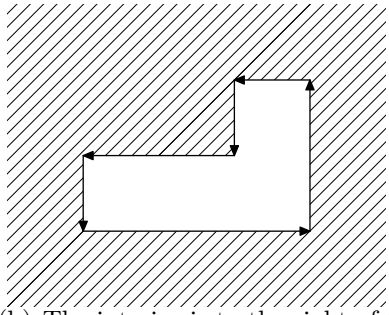
Figure 4.7: Previous mask shape representation example [20, 21, 24] : a fixed mask object and its variable mask objects. Vertices can present more than one times in the fixed mask object and its variable mask objects. The 45° shaded regions have positive convolution values, and the 135° shaded regions have negative convolution values. It is semantically equivalent to that of Figure 4.9 which uses our proposed new method.

p indicates whether it points to the positive direction (x or y) or the negative direction ($-x$ or $-y$). The per polygon Boolean variable c indicates whether the interior of the region is to the left or to the right of the edges of the polygon. O denotes the starting point of the first edge (e_0).

Figure 4.9 shows that the polygon in Figure 4.8 is segmented into many small segments, denoted as dashed arrows. In addition to the parameters $\{l_i, h_i, p_i\}$, one more parameter, d_i, should be used to describe the displacement of the segment from its original location. By applying nonzero d's to all the segments, the segments are shifted to the locations denoted as solid arrows. The vertices needed in lithography simulators can be computed based on this representation. Smaller number of vertices results in shorter lithography simulation runtime.



(a) The interior is to the left of the edges ($c = 1$).



(b) The interior is to the right of the edges ($c = 0$).

Figure 4.8: Rectilinear polygon representation. The interior region is shaded. Each edge is represented as a triplet $e_i = \{l_i, h_i, p_i\}$. The polygon is represented as $\{O, c, \{e_i | 0 \leqslant i < N\}\}$. In this example, $N = 6$.

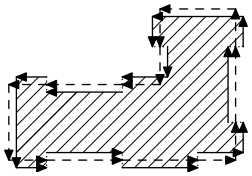


Figure 4.9: Segmented rectilinear polygon.

4.6.2 Segment Movement Scheme

Each segment can be moved based on the print contour information locally [24] or on the non local print contour information, e.g. the MEEF matrix based scheme [22].

We use the first approach, though our PV-OPC algorithm can be extended to use the second approach. We use the standard OPC segmentation and tagging strategy [24]. Each segment is moved based on the V-EPE metric $\langle \boldsymbol{E} \rangle$ at its control point. However, we update all the segment displacements at the same time, instead of updating them one at a time [24]. The details of the PV-OPC algorithm are shown in Algorithm 5. It is an iterative algorithm, where the constant C controls the edge movement step. The main difference

Algorithm 5 PV-OPC algorithm

```
1:  function PV-OPC(Non-touching polygons decomposed from the original design)
2:      Segment the polygons into movable edges and tag the middle points as their control points
3:      repeat
4:          updated ← false
5:          for all control points do
6:              compute the maximum aerial gradient direction
7:              store ⟨E⟩ along that direction
8:          for all edges do
9:              if |C⟨E⟩ · n| ⩾ manufacturing grid then
10:                 move the edge by −C⟨E⟩ · n (rounding to a multiple of manufacturing grids)
11:                 updated ← true
12:     until updated = false
```

compared to the conventional OPC algorithm is the objective function $\langle \boldsymbol{E} \rangle$ which incorporates the process-variation information. We could also use an-

other variation-EPE metric. Due to the analytical nature of our model and efficient table lookup, the complexity is the same as the conventional OPC, with just a slightly larger constant (as we shall show in the experimental results).

4.7 Results of Experiments

4.7.1 VLIM Calibration

We implemented VLIM in C++. We used PROLITH$^{\text{TM}}$ as our virtual fab and calibrated VLIM to the PROLITH$^{\text{TM}}$ simulation results. Note that the calibration method is generic enough to handle real fab data.

To calibrate the VLIM, we need to do PROLITH$^{\text{TM}}$ simulation on different patterns. In industrial lithography model test cases, there are many patterns, including lines, spaces and contacts, etc. For demonstration purpose, we only used four periodic line/space patterns, which have the same line width (65 nm), but different pitches (180, 300, 500 and 1000 nm).

For each pattern, we used VLIM to simulate at evenly sampled diffusion lengths d's (0, 2, 4,..., 20 nm). At each sampled d value, we also evenly sampled intensity threshold I_{th} and focus error z, such that $\left(\frac{I_{\text{th}}-I_{\text{th0}}}{\Delta I_{\text{th}}}\right)^2+\left(\frac{z}{\Delta z}\right)^2 < 1$, where $I_{\text{th0}} = 0.17$, $\Delta I_{\text{th}} = 0.02$ and $\Delta z = 80\,\text{nm}$. We computed the CD function $\text{CD}^{\boldsymbol{P}}(I_{\text{th}}, z, d)$ for pattern $\boldsymbol{P}$ using the method in Appendix C. The upper bound L, M and N are all set to 3 in our experiment.

We did PROLITH$^{\text{TM}}$ CD simulation on the four patterns at evenly sampled (E, Z) points in the region $\left(\frac{E-E_0}{\Delta E}\right)^2 + \left(\frac{Z-Z_0}{\Delta Z}\right)^2 < 1$, where $E_0 = 32.0\,\text{mJ/cm}^2$, $\Delta E = 1.0\,\text{mJ/cm}^2$, $Z_0 = 25\,\text{nm}$ and $\Delta Z = 75\,\text{nm}$. The PROLITH$^{\text{TM}}$ input parameters are summarized in Table 4.1. The fitting results are shown

Table 4.1: PROLITH^{TM} parameter summary.

90 nm node line ArF example - FC	
Photoresist thickness	232 nm
Brewer DUV 42C BARC	
BARC thickness	50 nm
BARC refraction index	$1.48 + 0.41i$
Silicon substrate	
Substrate refraction index	$0.883143 + 2.77779i$
Ideal bake mode	
Bake time	60 sec
Bake temperature	95 °C
Kirchhoff mask simulation mode	
Conventional partially coherent illumination	
Partial coherence	0.7
Wavelength	193 nm
Numerical aperture	0.8
Focal position relative to middle of resist	
Offset from the top	−40 nm
Positive numbers shift up	
Ideal PEB model	
PEB time	90 sec
PEB temperature	120 °C
Base surface contamination	
Relative contaminant concentration	0.001
Contaminant diffusion length	50 nm
User defined developer	
Develop time	20 sec
Vector image calculation model	
Full physics resist model	
Number of Exposure passes	1
Main speed factor	2
Speed factor for XY step	4
Speed factor for Z step	4
Source grid step size	0.0322581
X step size	2.25
Z step size	2.32

Table 4.2: Fitting results.

d	1.9484 nm
B	−0.666 471 nm
α_Z	22.7276 nm
β_Z	1.10935
α_E	11.5705 mJ/cm^2
β_E	3.301 49 mJ/cm^2
σ	0.71 nm

in Table 4.2. The parameter σ denotes the fitting error. It is only 0.71 nm for all test patterns, thus our VLIM is fairly accurate under process variations. It should be noted that as the number of test patterns increases, the fitting error may grow, which is expected for any phenomenological model. The calibration with real experimental data will also probably result in larger fitting error. However, to the first order, our VLIM is good enough to guide OPC or other mask/layout synthesis.

4.7.2 OPC results comparison

We implemented both the conventional OPC and the PV-OPC algorithms in C++. Our test layouts are the poly layers of an inverter and a NAND gate following 65 nm minimum and recommended design rules, named minINV, recINV, minNAND, recNAND. We use the nominal condition $I_{\text{th}} = 0.15$ and $z = 0$ nm for the conventional OPC and the distribution parameters $\mu_{I_{\text{th}}} = 0.15$, $\sigma_{I_{\text{th}}} = 0.007$, $\mu_z = 0$ nm and $\sigma_z = 80$ nm for the PV-OPC. The bias B is set to zero for both OPC algorithms.

We computed the CD mean $\langle \text{CD} \rangle$ and the CD variance Var(CD) at every 1 nm in the NMOS and PMOS active regions along the gate width direction. We then computed the averages of $\langle \text{CD} \rangle$ and Var(CD), denoted by

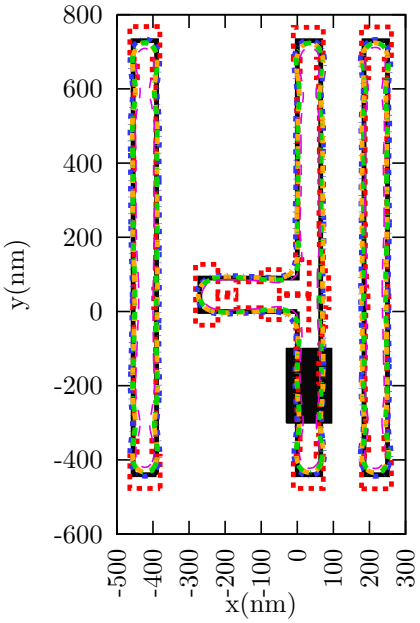
$\overline{\langle \mathrm{CD} \rangle}$ and $\overline{\mathrm{Var}(\mathrm{CD})}$, in each region. Table 4.3 shows the comparison between the results of the two OPC algorithms. $\overline{\langle \mathrm{CD} \rangle}$'s from the PV-OPC are much closer to 65nm than those from the conventional OPC. $\overline{\mathrm{Var}(\mathrm{CD})}$'s from the PV-OPC are comparable to those from the conventional OPC. Thus PV-OPC is more robust with respect to process variations.

Table 4.3: Post-OPC CD mean and EPE variance comparison.

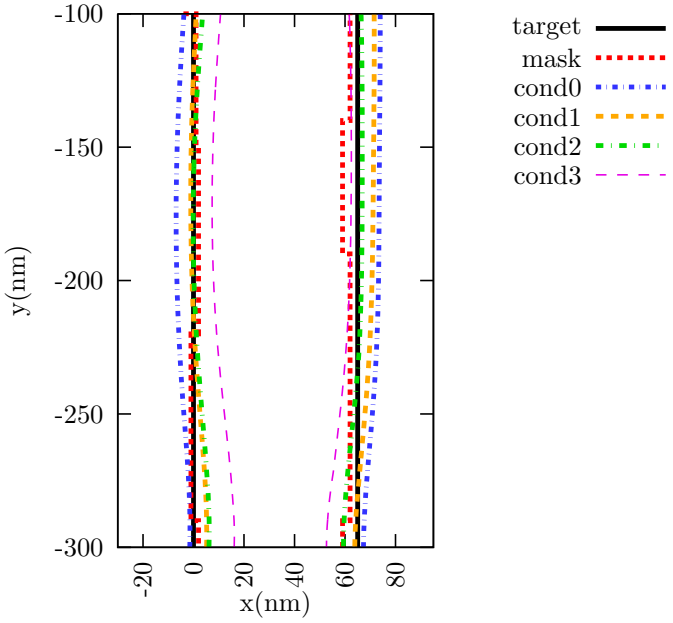
circuits	MOS type	average CD mean (nm)		average EPE variance (nm)	
		Conventional	PV-OPC	Conventional	PV-OPC
minINV	PMOS	58.83	66.28	3.56	3.77
	NMOS	58.89	66.75	3.27	3.47
recINV	PMOS	59.40	66.71	3.62	3.83
	NMOS	58.81	67.10	3.83	3.51
minNAND	PMOS	59.85	66.67	3.68	3.87
	NMOS	58.42	65.30	3.47	3.64
recNAND	PMOS	60.42	65.36	3.74	3.88
	NMOS	60.40	67.41	3.55	3.70

We show the results from both OPC algorithms for the poly layer layout of the inverter following the minimum design rules in Figure 4.10 and Figure 4.11. The NMOS region is indicated by the rectangles in Figure 4.10(a) and Figure 4.11(a). We also show the results from both OPC algorithms for the poly layer layout of the NAND gate following the recommended design rules in Figure 4.12 and Figure 4.13. One of the PMOS regions is indicated by the rectangles in Figure 4.12(a) and Figure 4.13(a). The targets, OPCed mask shapes and the printed contours at four process conditions (Table 4.4) are shown in these figures.

These figures show that the printed gate lengths tend to shrink due to

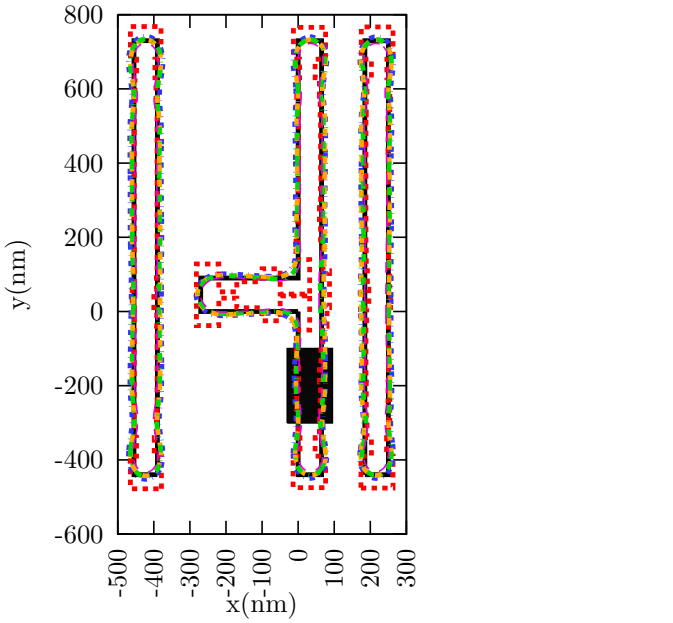


(a) Whole view

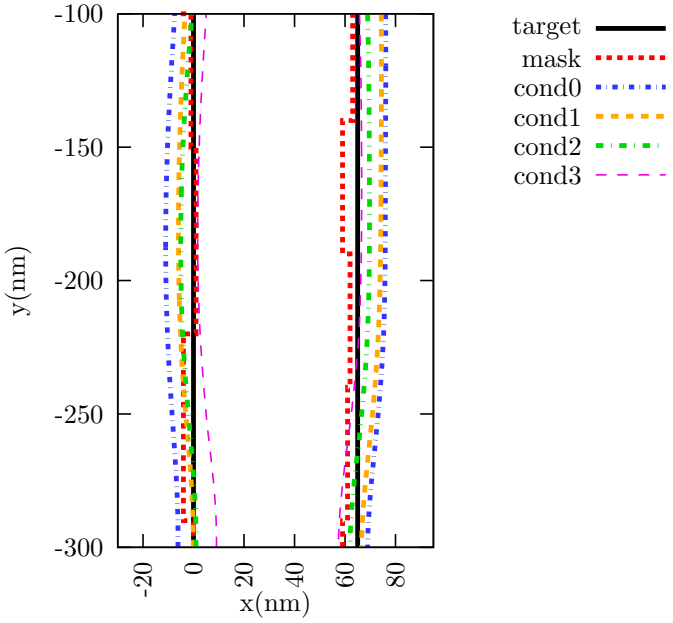


(b) NMOS region

Figure 4.10: Conventional OPC (inverter following the minimum design rules): $\overline{\mathrm{Var}(\mathrm{CD})}$ CD error is $-6.11\,\mathrm{nm}$. The four conditions labeled cond0 to cond3 are defined in Table 4.4.

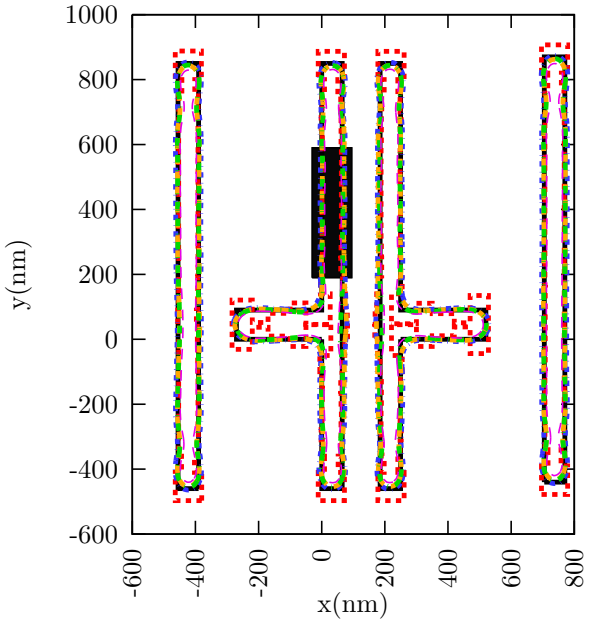


(a) Whole view

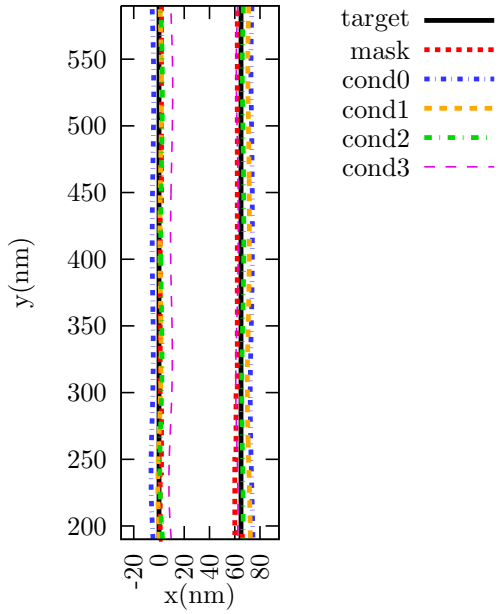


(b) NMOS region

Figure 4.11: PV-OPC (inverter following the minimum design rules): $\overline{\mathrm{Var(CD)}}$ error is 1.75 nm. The four conditions labeled cond0 to cond3 are defined in Table 4.4.

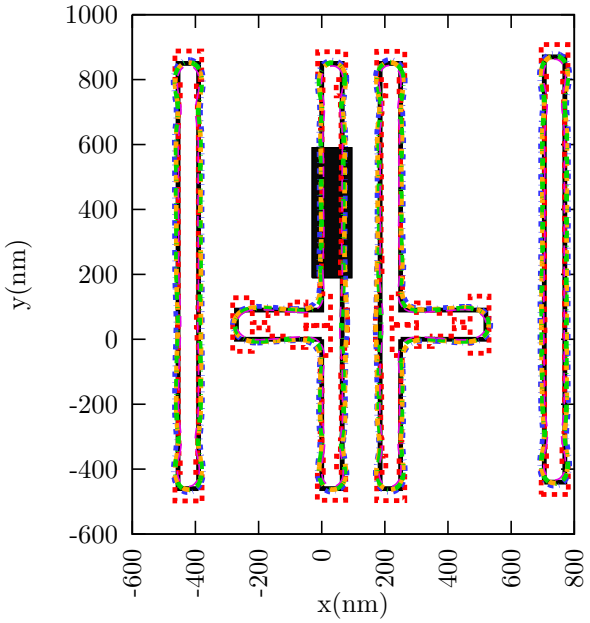


(a) Whole view

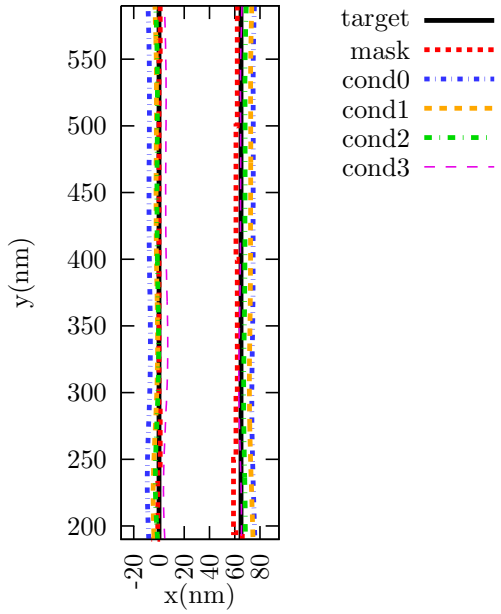


(b) PMOS region

Figure 4.12: Conventional OPC (NAND following the recommended design rules): $\overline{\mathrm{Var}(\mathrm{CD})}$ error is $-4.85\,\mathrm{nm}$. The four conditions labeled cond0 to cond3 are defined in Table 4.4.



(a) Whole view



(b) PMOS region

Figure 4.13: PV-OPC (NAND following the recommended design rules): $\overline{\mathrm{Var(CD)}}$ error is 0.36 nm. The four conditions labeled cond0 to cond3 are defined in Table 4.4.

Table 4.4: Four process conditions used in Figure 4.10, 4.11, 4.12 and 4.13.

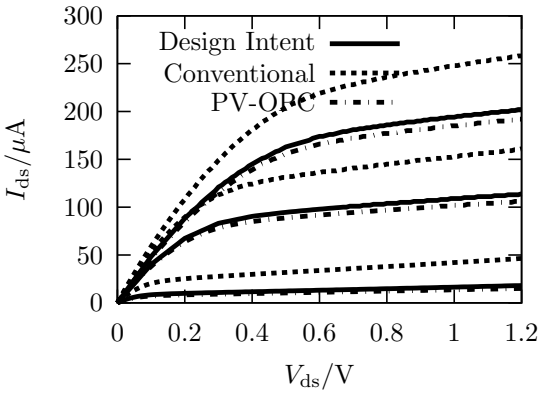
		I_{th}	
		0.143	0.157
z	0 nm	cond0	cond2
	80 nm	cond1	cond3

focus variations [1]. By comparing Figure 4.10 to Figure 4.11 and comparing Figure 4.12 to Figure 4.13, we can see that PV-OPC bias the edges toward the outside intelligently.

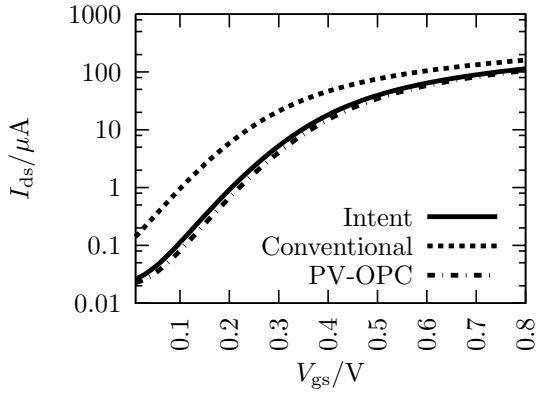
Figure 4.14 and Figure 4.15 show the robustness of PV-OPC with respect to the electrical characterizations. Figure 4.14(a) and Figure 4.15(a) show the NMOS and PMOS I-V curves at $V_{\text{gs}} = 0.4\,\text{V}$, $0.8\,\text{V}$ and $1.2\,\text{V}$, respectively. Figure 4.14(b) and Figure 4.15(b) show the NMOS leakage current as a function of V_{gs} at $V_{\text{ds}} = 1.2\,\text{V}$. The solid curves represent the design intent of the drawn layout. The dash curves represent the post-OPC expectation of the conventional OPCed mask considering lithography variation. The dash-dot curves represent the expectation of the variation-aware OPCed mask. Figure 4.14(a) and Figure 4.15(a) show that conventional OPC cannot make the I-V curve expectations the same as the design intent. However, our variation-aware OPC does a good job. Figure 4.14(a) and Figure 4.15(a) show that the sub-threshold leakage expectations of the conventional OPC results can be *4× bigger* than the design intent. Our variation-aware OPC algorithm effectively reduces the gap.

Table 4.5 shows the runtime and the total number of iterations of the conventional OPC algorithm and the PV-OPC algorithm. The PV-OPC run-

[1] Depending on the processes, it could expand as well.

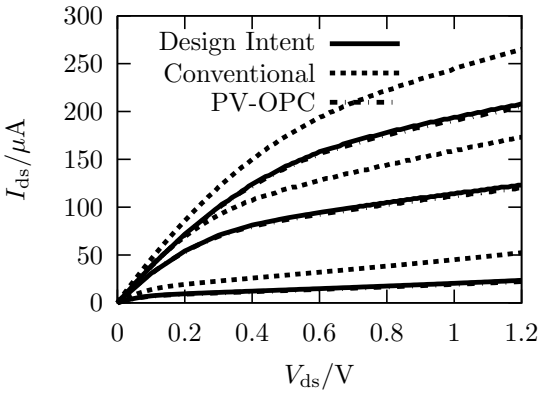


(a) I-V curves

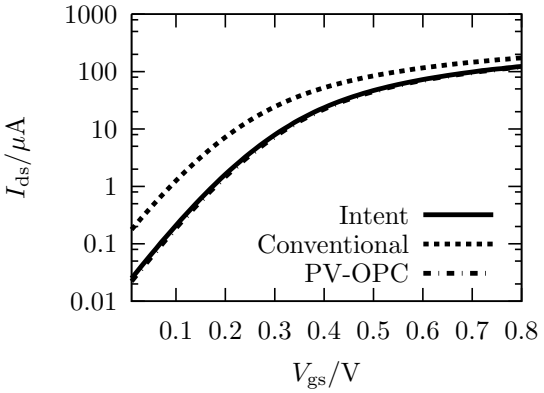


(b) Leakage (PV-OPC leakage curve is almost exact same as the design intent one)

Figure 4.14: NMOS electrical characterization of the inverter following 65 nm minimum design rules.



(a) I-V curves



(b) Leakage (PV-OPC leakage curve is almost exact same as the design intent one)

Figure 4.15: PMOS electrical characterization of the NAND gate following 65 nm recommended design rules.

time is only about 2-3× slower, which is very impressive considering that it explicitly incorporates the entire process window information.

Table 4.5: OPC Runtime Comparison

circuit	Conventional OPC		PV-OPC		Runtime Ratio
	Runtime	Iter#	Runtime	Iter#	
minINV	3.74 sec	42	10.87 sec	47	2.91
recINV	4.14 sec	42	10.91 sec	42	2.64
minNAND	5.57 sec	41	14.99 sec	44	2.69
recNAND	7.84 sec	52	17.46 sec	44	2.23

4.8 Summary

In this chapter, a new variational lithography model is derived and calibrated to the industry standard lithography simulation software PROLITH™. Based on this variational lithography model, a variational EPE metric is presented. A variational aware OPC algorithm is proposed based on this new metric. We show our implementation details of the PV-OPC algorithm. The PV-OPC algorithm is tested on some 65nm layouts. It obtains much more robust results than the conventional OPC in terms of both the geometric and electrical metric. The runtime increases only 2-3×, which is satisfactory for a true process-variation-aware OPC.

Chapter 5

Intensity Based Optical Proximity Correction

Regarding technology scaling, reducing OPC runtime while keeping good quality of result is very important. Conventional OPC algorithms are based on Edge Placement Error (EPE), which requires many intensity simulations, which take the majority of the OPC runtime. By making the OPC algorithm intensity based (IB-OPC) rather than EPE-based, we reduce the number of the intensity simulations and hence reduce the OPC runtime. We also provide an efficient intensity derivative computation method, which makes the new algorithm converge faster than the EPE-based algorithm. Our experimental results show a runtime speedup of more than 10× with comparable result quality as an EPE-based OPC.

Preliminary results of this work have been published in [106].

5.1 Introduction

OPC algorithms [24] modify mask shapes to compensate for the optical proximity effect due to the subwavelength lithography printing. It iteratively performs lithography simulation and corrects mask shapes based on the simulation results, until the printed contour is close enough to the target contour. OPC is a very time consuming process, which could take hours or days to finish for designs of 90 nm and 65 nm and could be even worse for 45 nm and 32 nm

technologies. Slow OPC runtime can affect product turnaround-time (TAT) adversely, which can dramatically reduce profits. Therefore, it is important to reduce OPC runtime. But the OPC result quality shall be maintained.

As we have mentioned in Section 1.1, OPC runtime can be improved by using parallel computation and dedicated hardware. Another aspect is to fine tune the OPC algorithms, the OPC algorithm parameters, and the recipes (e.g. segmentation and tagging). A few new convergence schemes are proposed to improve the convergence rate [65]. A neural network is proposed to give a better initial guess which will result in lower number of OPC iterations [50]. The runtime implications of a few OPC parameters are shown in [27], which point out the potential trade-off between various parameters for runtime. The number of simulation points for each simulation site can be optimized to reduce runtime [5]. Optimized multi-OPC recipes have also been proposed for SoC applications to reduce runtime and improve OPC accuracy [29]. Design information is used to determine the error (EPE) tolerance to reduce OPC runtime [48]. Cellwise OPC is done to reduce OPC runtime [69].

As opposed to the approaches above, in this chapter, we strive to improve OPC runtime from the algorithmic aspect. Conventional OPC algorithms are based on Edge Placement Error (EPE) [24], which is defined as the difference between the printed contour and the target contour. Note that we do not consider inverse lithography [73], since it is different from the polygon based OPC algorithms that we are considering here. Cobb et al. improved the EPE-based OPC algorithm using Mask Error Enhancement Matrix (MEEM) [23], where MEEM is the EPE sensitivity matrix with respect to changes in mask shapes. The authors claimed that the MEEM-based method is better than the EPE-based OPC for Alternating Phase-Shift Mask (AltPSM) or ex-

otic illuminations such as dipole illumination. However, the computation of neither EPE nor MEEM is cheap. EPE-based methods require many image intensity simulations, and MEEM-based methods need even more. To reduce the OPC runtime, we shall seek ways to reduce the number of lithography image simulations.

Based on a key observation that making EPE zero is equivalent to making the intensity equal to the intensity threshold computed from the photoresist model, we describe an intensity based OPC. Our experimental results demonstrate that it is faster than the conventional EPE-based OPC.

The main contributions of this chapter are as follows:

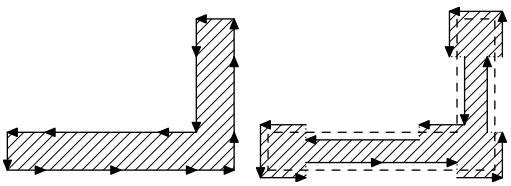
- We propose an intensity based OPC (IB-OPC) algorithm.
- We provide an efficient method for the intensity sensitivity computation.
- We demonstrate that IB-OPC achieves a significant runtime speedup, while maintaining the OPC result quality.

5.2 EPE Based OPC

We review the conventional EPE-based OPC algorithm in this section.

In VLSI layouts, there are usually only rectilinear polygons, which have only horizontal and vertical edges. The polygon edges are usually broken into smaller parts (Figure 5.1(a)), called segments, which can be shifted by OPC algorithms (Figure 5.1(b)).

Conventionally, the OPC algorithm is EPE based. The EPE is computed for some point (called a tag) on each segment. The segment is shifted



(a) (b)

Figure 5.1: (a) The edges of a rectilinear polygon are segmented. (b) The segments can be shifted by OPC algorithms.

for a fraction of the EPE to reduce the error. Algorithm 6 [23] shows the details, where C is the fraction of the EPE moved. As a vector, the symbol $\boldsymbol{x}_i$ denotes the shifts of all the segments.

Algorithm 6 EPE-based OPC algorithm

1: **function** EPE-OPC
2: Choose an appropriate initial solution $\boldsymbol{x}_0$
3: $i \leftarrow 0$
4: **repeat**
5: compute EPEs of all the tag points, denoted as $\boldsymbol{E}$
6: $\boldsymbol{x}_i \leftarrow \boldsymbol{x}_i - C\boldsymbol{E}$
7: $i \leftarrow i + 1$
8: **until** no edge has been changed

5.3 Intensity Based OPC Algorithm

5.3.1 Problem Formulation

EPE-based OPC tries to make EPE zero. However, the computation of EPE is time-consuming. In Figure 5.2, to compute the EPE near the tag point A, we need to simulate on the dots to find the printed contour. It may require many simulations before the contour is found. It may also need many simulations to know that the contour cannot be found.

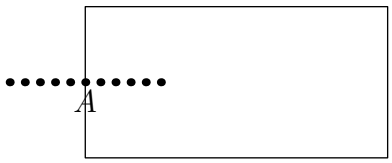


Figure 5.2: EPE computation requires many intensity simulations. The box is the target shape. To find the printed contour near the tag point A, we may need to simulate on many simulation points (dots).

Figure 5.3 shows that requiring EPE to be zero is equivalent to making the intensity at the target the same as the intensity threshold. This criterion enables us to simulate intensities at much fewer points (only on the tag point A in the example in Figure 5.2). And we have the following OPC problem formulation.

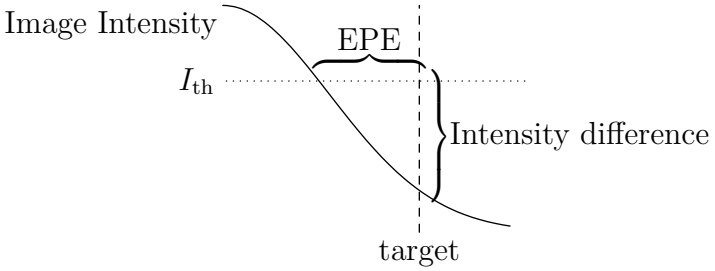


Figure 5.3: EPE is zero if and only if there is no intensity difference.

Formulation 1 (Intensity Based OPC Fomulation)**.** *Intensity Based OPC Algorithm matches the intensity on the target with the intensity threshold.*

Note that this formulation is very general: the intensity threshold is not necessarily a constant. To illustrate the main concept, we will focus on solving this problem for the constant threshold model.

Suppose there are N segments, which means the mask has freedom of degree N. We need exactly N constraints to uniquely solve the shifts of the

N segment vectors. We choose N tag points at the central of each segment. The intensity matches the threshold on these tag points.

Let us number the tagging points and the segments from 1 to N. I_i denotes the image intensity at the i-th tag point. The threshold at the i-th tag point is denoted as $I_{\text{th}i}$. Therefore, the mathematical formulation of IB-OPC is

$$I_i(d_1, d_2, \ldots, d_N) = I_{\text{th}i} \quad i = 1, 2, \ldots, N, \tag{5.1}$$

where d_i is the displacement of the i-th segment. Note that all $I_{\text{th}i}$ equal a single constant for the constant threshold model.

The system of equations (5.1) can be written compactly in the vector form as

$$\boldsymbol{A}(\boldsymbol{x}) = \boldsymbol{b} \tag{5.2}$$

where $\boldsymbol{x}$ denotes the entire vector of values d_i, $\boldsymbol{A}$ denotes the entire vector of function I_i and $\boldsymbol{b}$ denotes the entire vector of values $I_{\text{th}i}$.

5.3.2 The Algorithm

To solve the nonlinear system of equations (5.2), we adapt the Newton method discussed in [74] and modify it to fit our particular problem.

Suppose $\boldsymbol{x}$ is close enough to the solution to (5.2) and $\boldsymbol{A}(\boldsymbol{x})$ can be Taylor expanded in the neighborhood of $\boldsymbol{x}$ as

$$\boldsymbol{A}(\boldsymbol{x} + \delta\boldsymbol{x}) = \boldsymbol{A}(\boldsymbol{x}) + \boldsymbol{J} \cdot \delta\boldsymbol{x} + O(\delta\boldsymbol{x}^2), \tag{5.3}$$

where $\boldsymbol{J}$ is the *Jacobian* matrix of $\boldsymbol{A}$ at $\boldsymbol{x}$. By neglecting the second and higher order terms in (5.3) and setting $\boldsymbol{A}(\boldsymbol{x} + \delta\boldsymbol{x}) = \boldsymbol{b}$, we can solve the correction

term $\delta \boldsymbol{x}$ from

$$\boldsymbol{J} \cdot \delta \boldsymbol{x} = -\boldsymbol{A}(\boldsymbol{x}) + \boldsymbol{b}. \tag{5.4}$$

The solution $\delta \boldsymbol{x}$ of (5.4) shall be applied to $\boldsymbol{x}$ to approximate a better solution to (5.2). This process shall be repeated until convergence. However, according to our experience, the algorithm may not converge if the full Jacobian $\boldsymbol{J}$ is used. This is due to the high nonlinearity of the OPC problem. In addition, we do not need the exact value of $\delta \boldsymbol{x}$ in the iteration. Therefore, we use the diagonal $\boldsymbol{D}$ of $\boldsymbol{J}$ instead. Note that it is also cheaper to compute the diagonal matrix than to compute the full Jacobian matrix $\boldsymbol{J}$. This approximation can be intuitively understood because the intensity at any tag point is influenced mostly by the segment associated with it. We end up solving the following system of equations instead

$$\boldsymbol{D} \cdot \delta \boldsymbol{x} = -\boldsymbol{A}(\boldsymbol{x}) + \boldsymbol{b}. \tag{5.5}$$

Since $\boldsymbol{D}$ is a diagonal matrix, $\delta \boldsymbol{x}$ can be solved easily in (5.5).

It is intuitive to add the correction $\delta \boldsymbol{x}$ to $\boldsymbol{x}$ as

$$\boldsymbol{x}_{\text{new}} = \boldsymbol{x}_{\text{old}} + \delta \boldsymbol{x}, \tag{5.6}$$

and to iterate until it converges. However, this method would not converge if the initial guess is not sufficiently close to the solution. In current OPC algorithms, the initial guess is usually chosen to be $\boldsymbol{x} = \boldsymbol{0}$, which could be far away from the solution. Therefore, we need to make our solution scheme converge globally even if the initial guess is far from the solution.

A reasonable strategy is to go to some point $\boldsymbol{x}_{\text{new}}$

$$\boldsymbol{x}_{\text{new}} = \boldsymbol{x}_{\text{old}} + \lambda \boldsymbol{p}, 0 \leqslant \lambda \leqslant 1 \tag{5.7}$$

along the direction of $\boldsymbol{p} = \delta\boldsymbol{x}$. In addition, $f = \frac{1}{2}\boldsymbol{F}\cdot\boldsymbol{F}$ must always decrease, where $\boldsymbol{F}(\boldsymbol{x}) \equiv \boldsymbol{A}(\boldsymbol{x}) - \boldsymbol{b}$ and the factor $\frac{1}{2}$ is for later convenience. We will show below how to approximately find λ so that $f(\boldsymbol{x}_{\text{old}} + \lambda\boldsymbol{p})$ decreases.

Let us define

$$g(\lambda) = f(\boldsymbol{x}_{\text{old}} + \lambda\boldsymbol{p}). \tag{5.8}$$

We want to find an approximate minimum of $g(\lambda)$ $(0 \leqslant \lambda \leqslant 1)$. We could approximate $g(\lambda)$ as a parabola [74], which requires three parameters to be uniquely determined. We choose the following two methods to approximate $g(\lambda)$, which trade-off speed and accuracy differently.

- We set the parabola be $g(0)$ and $g(1)$ at $\lambda = 0$ and 1 and the derivative at $\lambda = 0$ be $g'(0)$. We will show below that this method does not need an additional evaluation of $\boldsymbol{F}$ as we shall show below, which is faster.

  The derivative of $g(\lambda)$ is

  $$g'(\lambda) = \nabla f \cdot \boldsymbol{p} = (\boldsymbol{F}\cdot\boldsymbol{J})\cdot(-\boldsymbol{D}^{-1}\cdot\boldsymbol{F}). \tag{5.9}$$

  Since we only need a fast estimate of $g'(0)$, we again approximate $\boldsymbol{J}$ as $\boldsymbol{D}$ and get

  $$g'(\lambda) \approx -\boldsymbol{F}\cdot\boldsymbol{F}. \tag{5.10}$$

  Therefore, we have

  $$g'(0) \approx -2g(0), \tag{5.11}$$

  which does not require an additional evaluation of $\boldsymbol{F}$.

With $g(0)$, $g'(0)$ and $g(1)$ available, we model $g(\lambda)$ as a parabola:

$$g(\lambda) \approx (g(1) - g(0) - g'(0))\lambda^2 + g'(0)\lambda + g(0) \tag{5.12}$$

We can easily find that its minimum is taken at

$$\lambda_{\min} = -\frac{g'(0)}{2(g(1) - g(0) - g'(0))} = \frac{g(0)}{g(0) + g(1)}. \tag{5.13}$$

- We make the parabola be $g(0)$, $g(1/2)$ and $g(1)$ at $\lambda = 0$, $1/2$ and 1. It can be solved that the parabola passing through these three points is

$$f(x) = ax^2 + bx + c, \tag{5.14}$$

where

$$\begin{aligned} a &= 2(g(0) - 2g(1/2) + g(1)) \\ b &= -3g(0) + 4g(1/2) - g(1) \\ c &= g(0). \end{aligned}$$

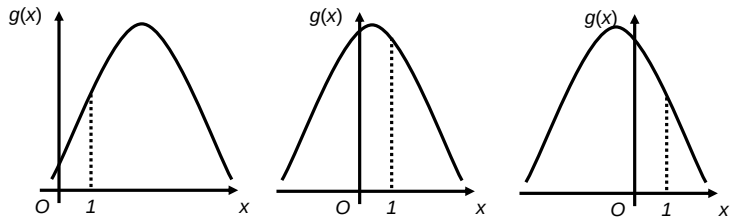
Figure 5.4 shows that the minimum $g(\lambda)$ is always at either 0 or 1, when $a \leqslant 0$. When $a > 0$, the minimum $g(\lambda)$ is achieved

$$\lambda_{\min} = -\frac{b}{2a} = \frac{3g(0) - 4g(1/2) + g(1)}{4(g(0) - 2g(1/2) + g(1))}, \tag{5.15}$$

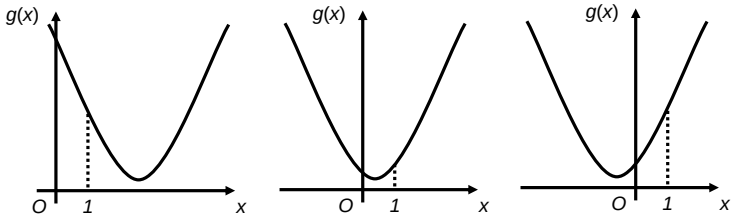
if $0 < -\frac{b}{2a} < 1$. Otherwise, $\lambda_{\min}$ is 1, if $-\frac{b}{2a} \geqslant 1$; or $\lambda_{\min}$ is 0, if $-\frac{b}{2a} \leqslant 0$.

The above $\lambda_{\min}$ is an approximation. We eventually need to check that $g(\lambda_{\min})$ is indeed less than $g(0)$. Otherwise, we do not accept it and the algorithm stops.

We list the IB-OPC algorithm in Algorithm 7 for completeness. This algorithm is fast for three reasons:



(a) When $a \leqslant 0$, the minimum is always at either 0 or 1.



(b) When $a > 0$, the minimum can be achieved at $\lambda = 1$, $-\frac{b}{2a}$ and 0.

Figure 5.4: Parabola approximation of $g(\lambda)$.

1. It is intensity based, which requires much less intensity simulation compared with the conventional EPE-based method.

2. The intelligent sensitivity computation method enables it to compute $\boldsymbol{D}$ rapidly.

3. Newton method makes the algorithm converge in fewer iterations.

5.3.3 Extension to the Variable Threshold Model

In practice, the threshold is determined based on the variable threshold model (VTR) [41]. Based on Algorithm 7, we could have the IB-OPC algorithm for the variable threshold model (Algorithm 8).

Algorithm 7 IB-OPC algorithm

function IB-OPC
    Choose an appropriate initial solution $\boldsymbol{x}_0$
    $i \leftarrow 0$
    repeat
        Solve $\delta \boldsymbol{x}_i$ based on the equation

$$\boldsymbol{D}(\boldsymbol{x}_i) \cdot \delta \boldsymbol{x}_i = -\boldsymbol{A}(\boldsymbol{x}_i) + \boldsymbol{b} \tag{5.16}$$

        Compute $\lambda_{\min}$ based one of the two parabola approximations discussed above
        if $g(\lambda_{\min}) \geqslant g(0)$ **then break**
        $\boldsymbol{x}_{i+1} \leftarrow \boldsymbol{x}_i + \lambda_{\min} \delta \boldsymbol{x}_i$
        $i \leftarrow i + 1$
    until $\boldsymbol{x}_i = \boldsymbol{x}_{i-1}$ // stop if nothing changes

Algorithm 8 IB-OPC algorithm with Variable Threshold Model

function IB-OPC w/ VTR
    Choose an appropriate initial solution $\boldsymbol{x}_0$
    repeat
        Determine $I_{\text{th}i}$ based on current the intensity profile using VTR
        call IB-OPC algorithm (Algorithm 7)
    until Converges

5.4 Lookup-Table Method for Intensity and Intensity Sensitivity Computation

When we use the Newton method to solve the system of equations (5.2), we implicitly assume that $\boldsymbol{A}(\cdot)$ is a continuous function of $\boldsymbol{x}$. It means that the image intensity at any point shall be a continuous function of the shifts of all the mask segments. This property guarantees that a small mask change results in only small intensity changes.

The intensity can be computed by two table-lookup methods; that is, the edge based method and the vertex based method [20, 24, 109, 110]. Below, we will briefly review these two methods. Readers may refer to [20, 109, 110] for more details. We will show that the vertex based method preserves the property of the continuity but the edge based method does not. Therefore, we will use the vertex based method for the intensity simulation.

Because of this property of the continuity, we can well define the intensity derivative with respect to mask segment shifts, as well as we provide an efficient method to compute it. We use this method to compute the diagonal $\boldsymbol{D}$ in the IB-OPC algorithm in Section 5.3.

5.4.1 Requirement of Continuous-Intensity Property of Lookup-Table Methods

The vertex based and edge based lookup-table methods are two methods for the convolution computation $(Q_n **F)$ in (1.13). The image intensity can be computed easily after the convolutions are computed. The mask shapes need to be clipped to the support region to save unnecessary computation.

Figure 5.5 shows an example for the vertex based lookup-table method. After clipping the mask shape BCEF (which has four segments, BC, CE, EF

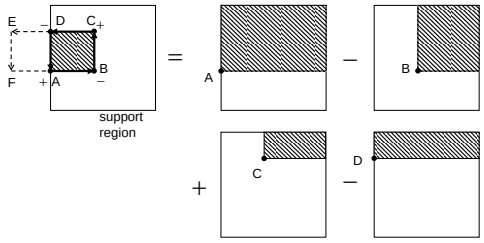


Figure 5.5: The vertex based lookup-table method. The five big squares are support regions. The shaded areas represent the convolution regions.

and FB) to the support region, we get the shape ABCD. The convolution of the shape ABCD can be decomposed into the 4 convolutions associated with the 4 vertices (A, B, C and D). The "+" and "−" signs represent whether the convolution associated with a vertex shall be added to or subtracted from the final convolution value.

The sign, "+" or "−", do not change for any vertex, no matter how the segments are shifted. For example, in Figure 5.6, we shift the segments AB and CD so that AB is above CD. But "+" is still associated with B and

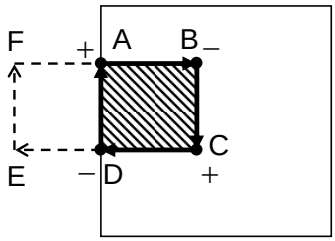


Figure 5.6: The vertex based lookup-table method for a modified shape.

D, and "−" is still associated with A and C. Note that the shape ABCD in Figure 5.5 is counterclockwise and the shape ABCD in Figure 5.6 is clockwise. Suppose ABCD in both figures occupy the same area. It is easy to see that the convolutions of the two shapes are a pair of opposite numbers. Therefore, the

intensity computed in (1.13) shall be the same in this case. A zero intensity is achieved when AB and CD collapse. Therefore, the intensity is continuous with respect to the mask changes in the vertex based method as shown in Figure 5.7.

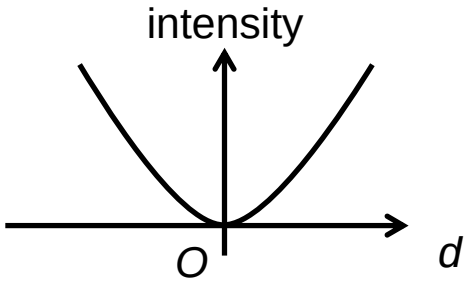


Figure 5.7: The intensity is continuous with respect to the mask changes in the vertex based method. d is the distance between AB and CD, which is positive when AB is below CD, and is negative otherwise.

Now, let us consider the edge based lookup-table method. We define the convention such that segments always go counterclockwise so that the edge based method and the vertex based method give the same convolution result for the case in Figure 5.5. However, they give different results: the shaded area is different in Figure 5.6 and the left figure in Figure 5.8. We could make the vertex based method and the edge based method give the same result for the case in Figure 5.6 if we change the definition such that the segments go clockwise rather than counterclockwise. But according to this new definition, the convolution for the right figure of Figure 5.8 would not be correct. The problem of the edge based method is that it cannot distinguish the difference between the two cases in Figure 5.8. In Figure 5.9, we show the image intensity as a function of the difference between AB and CD for the left figure of Figure 5.8. Obviously, there is an intensity discontinuity when AB

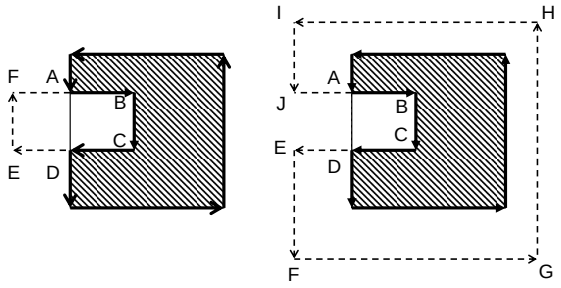


Figure 5.8: The edge based lookup-table method. The left figure shows the convolution region for the example BCEF in Figure 5.6. The right figure shows an example BCEFGHIJ which gives the same convolution result.

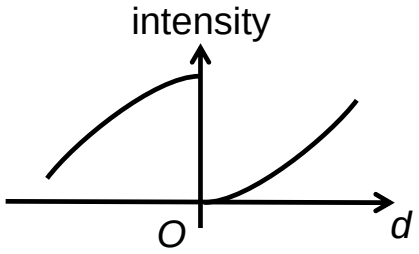


Figure 5.9: The intensity is not always continuous with respect to the mask changes in the edge based method. d follows the same convention as in Figure 5.7.

and CD collapse ($d = 0$). Due to the continuous property of the vertex base method, we choose it to compute the intensities.

5.4.2 Efficient Intensity Sensitivity Computation Method

A naive way of intensity sensitivity computation is to perturb the mask shapes a little and resimulate the intensity. The intensity change is proportional to the intensity sensitivity. However, this method requires two intensity simulations, which is slow. We derive an intelligent way below for faster sensitivity simulations.

The intensity change due to a small mask change ΔF_i can be derived from (1.13) as

$$I[F + \Delta F_i] - I[F] = \sum_{n=0}^{p-1} 2\sigma_n (Q_n \ast\ast F)(Q_n \ast\ast \Delta F_i), \tag{5.17}$$

where ΔF_i is due a shift of the i-th segment. Suppose the i-th segment is horizontal. ΔF_i can be expressed as

$$\begin{aligned} \Delta F_i = &\operatorname{boxcar}(x; a_i, b_i) \\ &\times \operatorname{boxcar}(y; y_i + d_i, y_i + d_i + \Delta d_i), \end{aligned} \tag{5.18}$$

where a_i and b_i denote the two ends of the i-th segment, d_i denotes the shift of that segment and Δd_i denotes its additional small shift. And boxcar is a function defined as

$$\operatorname{boxcar}(x; a, b) = \begin{cases} 1, & \text{if } a \leqslant x \text{ and } x \leqslant b \\ 0, & \text{otherwise.} \end{cases} \tag{5.19}$$

By taking the limit $\Delta d_i \to 0$, the sensitivity of the image intensity can

be derived as

$$\frac{\partial I}{\partial d_i} = \sum_{n=0}^{p-1} 2\sigma_n (Q_n \ast\ast F) \times (Q_n \ast\ast \text{boxcar}(x; a_i, b_i)\delta(y - (y_i + d_i))), \tag{5.20}$$

where $\delta(\cdot)$ is the Dirac delta function. The conventional lookup table method [109] can be used to compute $(Q_n \ast F)$ since the convolution is a linear operator. By using the same property, we can construct another table to compute $(Q_n \ast \ast(\text{boxcar}(x; a_i, b_i)\delta(y - (y_i + d_i)))$. In this case, only two table lookups (for the point (x, a_i) and the point (x, b_i)) are needed to compute the convolution.

We compare the complexity of this new method with the naive method, which computes the intensities twice. Suppose we need to look up the table for M times to compute the intensity: using the naive method, we need $2M$ table lookups to get the intensity sensitivity. However, we only need $2 + M$ table lookups to get the sensitivity in the intelligent method. Usually M is much larger than 2. Therefore, we get a speedup of $2\times$ using the new method.

The sensitivity for vertical segments can be computed using the table lookup method as well.

5.5 Results of Experiments

We implemented the EPE based OPC algorithm (Algorithm 6) and our IB-OPC algorithm (Algorithm 7) in C++. The following experiments were performed on a 2.66 GHz Intel Core 2 Duo Linux machine. We used 1 nm mask grid size (scaled to wafer). We used the conventional partially coherent illumination with $\sigma = 0.7$, the numerical aperture NA = 0.8, the wavelength $\lambda = 193$ nm. We used 10 kernels with the interaction radius of 600 nm. The

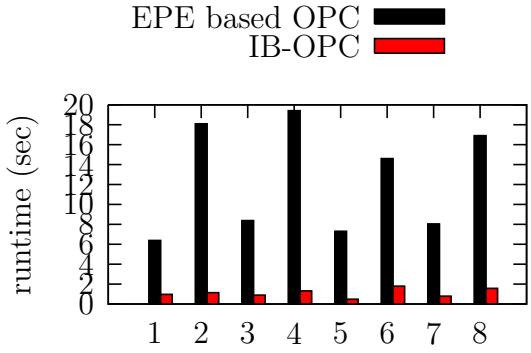
intensity threshold was 0.15. The 8 test cases are from a 65 nm technology poly layout. We used 50 nm and 100 nm segment lengths. Table 5.1 shows the number of segments.

Table 5.1: Number of segments for all the test cases.

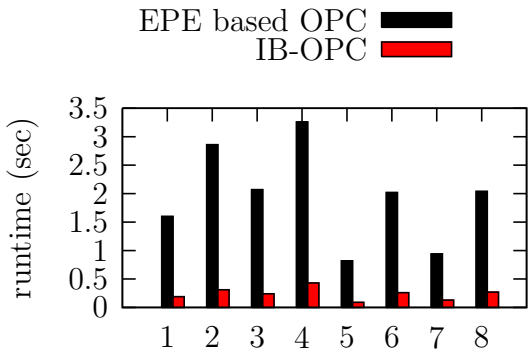
Case #	Segment length	
	50 nm	100 nm
1	1168	592
2	1568	800
3	1328	672
4	1760	896
5	832	416
6	1280	640
7	936	472
8	1440	720

We tried Algorithm 6 with $C = 0.10, 0.15, 0.20, 0.25, 0.30, 0.35$ and 0.40 on all the test cases for both 50 nm and 100 nm segment lengths. According to our experiments, the algorithm does not converge for all the test cases if $C = 0.35$ or 0.40. In general, the EPE becomes smaller as C increases, as long as the algorithm converges. Therefore, we chose C as 0.30 to make EPE small while still achieving convergence in the following experiments.

Figure 5.10 shows the runtime comparison between our EPE based OPC and IB-OPC with the first parabola approximation (that is, using (5.13) in the search algorithm) for 50 nm and 100 nm segment lengths on the test cases. Figure 5.11 shows the number of iterations comparison.

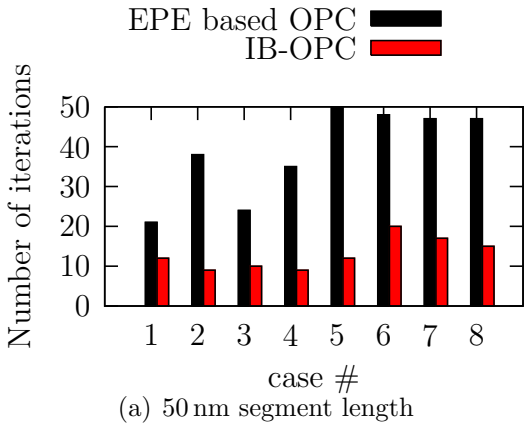


(a) 50 nm segment length

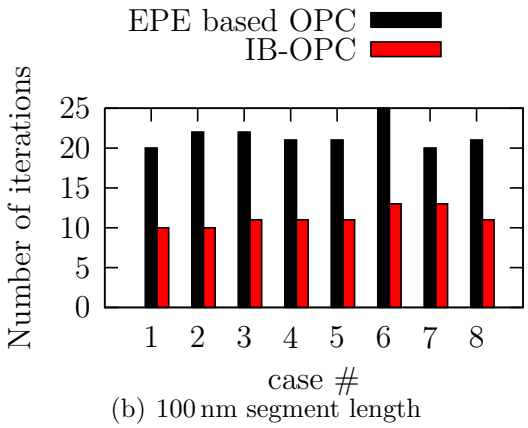


(b) 100 nm segment length

Figure 5.10: Runtime comparison.



(a) 50 nm segment length



(b) 100 nm segment length

Figure 5.11: Number of iterations comparison.

The runtime improvement can be separated into two parts as

$$\begin{aligned} &\text{Total runtime improvement} \\ =&\text{Improvement of the number of iterations} \\ &\times \text{Improvement of the runtime per iteration.} \end{aligned} \tag{5.21}$$

Table 5.2 shows the averages of three terms in (5.21) in our experiments. The runtime per iteration improves because IB-OPC does not compute EPE

Table 5.2: Average runtime improvement and number of iterations improvement.

Improvement	Segment length	
	50 nm	100 nm
Total runtime	11.5	8.2
Number of iterations	3.1	1.9
Runtime per iteration	3.7	4.3

and save unnecessary intensity simulations. The Newton method reduces the number of iterations. We can see that the number of iterations improvement increases as the segment length decreases. The total runtime speedup from IB-OPC can be up to 11.5× for 50 nm segment length, which is a significant improvement. As technology scales down, we would expect the segment length to be even smaller, in which case IB-OPC could give even more runtime improvement compared with the EPE-based OPC. Besides the improvement of runtime, we can see the correction results of IB-OPC is comparable with those of EPE based OPC from Table 5.3 and 5.4.

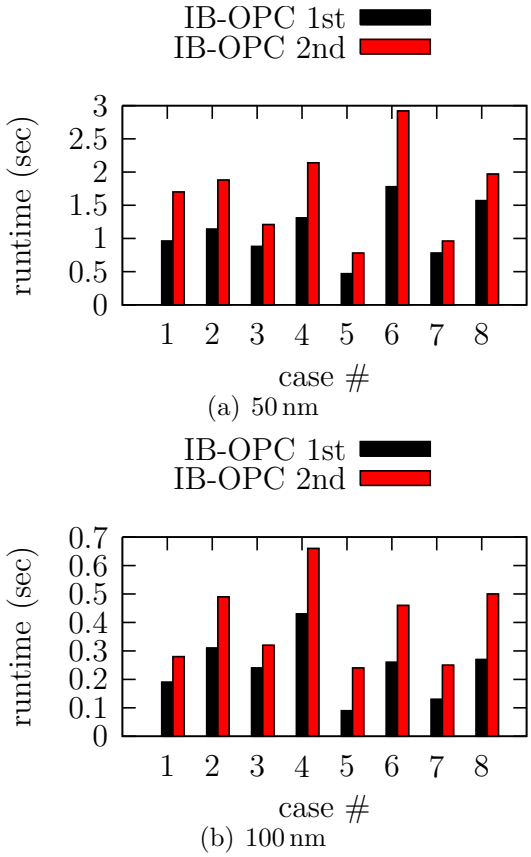
We also experimented with the IB-OPC algorithm using the second parabola approximation; that is, using (5.15) in the search algorithm. Figure 5.12 shows the runtime comparison between the IB-OPC algorithm using these two approximations. We can see that the IB-OPC algorithm using the

Table 5.3: EPE statitics (EPE based OPC).

Case #	Segment length			
	50 nm		100 nm	
	$\mu_{\|\mathrm{EPE}\|}$	$\sigma_{\|\mathrm{EPE}\|}$	$\mu_{\|\mathrm{EPE}\|}$	$\sigma_{\|\mathrm{EPE}\|}$
1	1.73	0.97	1.78	0.91
2	1.70	0.97	1.72	0.91
3	1.61	1.00	1.62	0.83
4	1.62	1.00	1.60	0.97
5	1.66	1.01	1.78	0.92
6	1.77	0.97	1.76	0.91
7	1.60	1.01	1.80	0.91
8	1.68	0.96	1.75	0.86

Table 5.4: EPE statitics (IB-OPC 1st).

Case #	Segment length			
	50 nm		100 nm	
	$\mu_{\|\mathrm{EPE}\|}$	$\sigma_{\|\mathrm{EPE}\|}$	$\mu_{\|\mathrm{EPE}\|}$	$\sigma_{\|\mathrm{EPE}\|}$
1	2.12	1.69	2.15	1.19
2	2.77	2.52	1.94	1.20
3	2.09	1.62	1.78	1.14
4	2.58	2.37	1.95	1.16
5	2.03	1.56	2.20	1.21
6	2.11	2.00	2.18	1.16
7	1.80	1.34	2.14	1.15
8	2.09	2.06	2.12	1.09



(a) 50 nm

(b) 100 nm

Figure 5.12: Runtime comparison of IB-OPC with the two parabola approximations.

second parabola approximation is slower than the one with the first parabola approximation, but it gives better OPC result quality as shown in Table 5.4 and 5.5.

Table 5.5: EPE statistics (IB-OPC 2nd).

Case #	Segment length			
	50 nm		100 nm	
	$\mu_{\|\mathrm{EPE}\|}$	$\sigma_{\|\mathrm{EPE}\|}$	$\mu_{\|\mathrm{EPE}\|}$	$\sigma_{\|\mathrm{EPE}\|}$
1	1.50	1.10	1.14	0.82
2	2.25	2.24	0.99	0.73
3	1.88	1.56	1.37	1.04
4	2.18	2.20	1.02	0.76
5	1.65	1.42	1.15	0.83
6	1.69	1.44	1.20	0.80
7	1.56	1.35	1.13	0.77
8	1.81	1.74	1.16	0.77

5.6 Summary

Conventional OPC algorithms are EPE based, which requires many image intensity simulations. To reduce runtime, we proposed an intensity based OPC algorithm, which requires fewer intensity simulations. The speed of the algorithm is further increased up by a variant of the Newton method for fast convergence, which computes the intensity sensitivity using an intelligent method. Our experiments show significant improved runtimes and good OPC result quality.

Chapter 6

Topologically Invariant Pixel Based Optical Proximity Correction

Currently, there are two advanced OPC approaches — the model-based OPC (MB-OPC) and the inverse lithography technology (ILT). MB-OPC generates masks which are less complex compared with ILT. But ILT produces much better results than MB-OPC in terms of contour fidelity because ILT is a pixel based method. Observing that MB-OPC preserves the mask shape topologies, which leads to a lower mask complexity, we can combine the strengths of both methods — the topology invariant property and the pixel based mask representation. This topologically invariant pixel based OPC (TIP-OPC) paradigm fills the critical hole in the OPC landscape and potentially has many new applications. Our work includes the lithography friendly mask topologically invariant operations, the efficient Fast Fourier Transform (FFT)-based cost function sensitivity computation and the TIP-OPC algorithm. The results of our experiments show that TIP-OPC can achieve much better post OPC contours compared with MB-OPC, while maintaining the mask shape topologies.

Preliminary results of this work have been published in [107].

6.1 Introduction

The most widely used OPC is Model-Based OPC (MB-OPC) [20, 21, 24], as discussed in Chapter 4 and 5. However, the OPCed result quality is subject to the OPC recipes (rules for segmenting and tagging), which may be very complex and hard to tune [17]. Also, the parametrized polygons are not flexible enough to represent any possible shape, which could result in contour fidelity degradation.

Alternatively, inverse lithography technology (ILT) [1, 39, 44, 51, 60, 73, 91] represents mask shapes as pixel images, which allows more flexible mask shape modifications. This approach could result in better image fidelity compared with MB-OPC. It has also been demonstrated that hardware acceleration and parallel computation can improve the runtime significantly such that this method can be applied to a full chip in practice [66]. However, all the published ILT formulations [39, 44, 72, 73] do not constrain the mask shapes explicitly. The ILT algorithms add Sub-Resolution Assist Features (SRAF) by models. But they may add too many SRAFs, which result in great mask complexity. As an example, Figure 6.1 shows the target patterns and the mask pattern generated by ILT in [73]. In addition, these approaches do not guarantee the minimum size of SRAFs, while small SRAFs may induce mask inspection problems.

MB-OPC does not add SRAFs, which maintains the mask shape topology. We call the OPC algorithm that maintains mask topologies "topologically invariant OPC (TI-OPC)". ILT clearly belongs to the other category, "topological variant OPC (TV-OPC)". Figure 6.2 shows a cartoon picture of the difference between TI-OPC and TV-OPC.

(a) Target (b) Mask

Figure 6.1: Mask patterns generated from ILT can be extremely complicated with many small features and SRAFs [73].

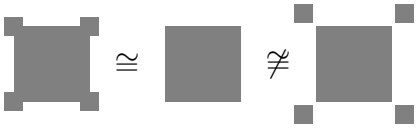


Figure 6.2: A cartoon picture of the TI-OPC (left) and TV-OPC (right) results. The center one is the target. The left mask has the same topology as the target, while the right one is not topologically equivalent to the target.

It is beneficial to combine the advantage of MB-OPC (topological invariance) and the advantage of ILT (pixel based). We call the new paradigm "topology invariant pixel based OPC (TIP-OPC)". We summarize the features of TIP-OPC below:

1. It has been found to be superior compared to the MB-OPC in terms of print contour fidelity. Since the mask is in pixel format, TIP-OPC has more freedom to modify masks in comparison with MB-OPC. Therefore, it can serve as a benchmark to evaluate the printability of designs using MB-OPC.

2. Since TIP-OPC maintains the mask shape topologies, the mask shapes produced by TIP-OPC will be less complex than ILT.

3. It can also be used in combination with ILT. In ILT, the mask trans-

mission value is solved as a real number, and eventually it has to be rounded to 0 or 1 (for a binary mask), which leads to errors in contours. TIP-OPC can be used to refine the mask and correct the contour errors.

4. Since TIP-OPC is pixel based and since any mask can be converted to pixel format easily, it can serve as an engineering change order (ECO) OPC. This is useful during the development stage of a new process, where the lithography simulation model might change slightly or we may want to do OPC incrementally on an already OPCed mask to transfer designs between fabs. MB-OPC is not easy to modify for this need.

5. Due to the fact that TIP-OPC is an extension of MB-OPC, it can be combined with conventional rule based SRAF insertion easily.

The first two items are important for Design-For-Manufacturability (DFM) analysis. That is, if the designs are not friendly to TIP-OPC, we would know they are definitely not good for MB-OPC, which means designs would have to be modified. Otherwise, even if the designs do not have good contour fidelity using MB-OPC, it could be due to imperfection in the OPC recipes. In this case, we should look for better recipes, which may save unnecessary design modifications.

The main contributions of this chapter are as follows:

- We introduce a new topologically invariant paradigm for pixel based OPC, which provides a trade-off between OPC quality and mask complexity which is not easy in the current methodologies, such as MB-OPC and ILT.

- TIP-OPC has finer control over masks than MB-OPC, and as such it can achieve better contour fidelity.
- Our algorithm is enabled by efficient techniques on topologically invariant mask operations and fast FFT-based cost function sensitivity computation.
- We derive, for the first time, the optimal tile size for the dense simulation method, which is used by TIP-OPC.
- TIP-OPC can be used with SRAF insertion algorithms seamlessly.

6.2 Analysis of the Complexities of Sparse and Dense Simulation Methods

In this section, we will compare the sparse simulation method (the lookup-table-method of Section 4.4) and the dense simulation method (shall be discussed below), which are used in MB-OPC and pixel-based OPC (e.g., ILT and TIP-OPC), respectively.

Besides the lookup table based image simulation method, the image intensity $I(x, y)$ can be computed in the frequency domain using a FFT as

$$I(x,y) = \sum_{p=0}^{P-1} \sigma_p \Big| \mathcal{F}\big[H_p \mathcal{F}^{-1}[m]\big] \Big|^2, \tag{6.1}$$

where $\mathcal{F}$ and $\mathcal{F}^{-1}$ are the Fourier and inverse Fourier transform operators, H_p's are kernels in the frequency domain and m is the mask.

As technology scales, it becomes important to perform lithography simulation on more locations to avoid printing any unnecessary features [15, 16,

19]. Simulating densely is also preferred to take advantage of the latest photoresist modeling advancement [42]. Lithography simulations are performed densely in pixel-based OPC since achieving better contour fidelity is the goal of pixel-based OPC.

Since in both methods, the images are only simulated on grid points, we analyze their time complexities by studying a chip discretized to a grid of size $L \times L$. We list the following terms used in sparse simulation method complexity:

- L^2 is the number of points on the simulation grid.
- s is the percentage of grid points where simulations are performed.
- v is the ratio between the total number of polygon vertices and the total number of mask pixels.
- K^2 is the region where the spatial kernels ($h_p = \mathcal{H}$) are non-zero.

In the sparse simulation method, precomputed convolution values for primitive shapes are saved in a lookup table. The mask shape m is first decomposed into these primitive shapes. Then, the convolution values are retrieved from the table. Due to the linearity of convolution, by adding or subtracting these values, the convolution of the mask shape can be obtained. Figure 6.3 shows an example mask near the point O. Only the mask shape in the support region of O is needed to compute the intensity at O. Therefore, in Figure 6.4, the mask is truncated to the support region. The truncated mask is decomposed into multiple polygons. The convolution of one polygon can be computed by looking up the table as shown in Figure 6.5. In Figure 6.5,

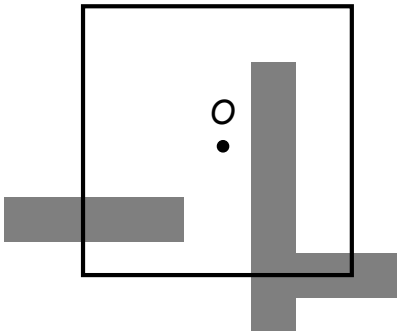


Figure 6.3: An example mask near the point O. The regions represent the mask shapes. The square around O is the support region.

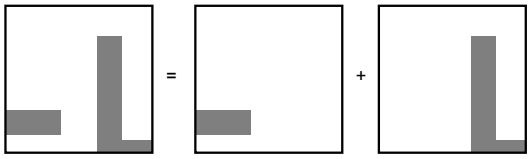


Figure 6.4: The mask is truncated to the support region of O. The mask is decomposed into multiple polygongs.

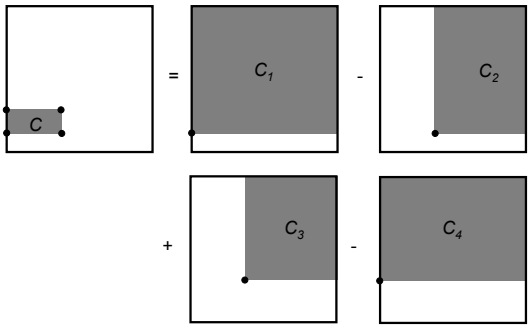


Figure 6.5: The convolution of a polygon can be computed by looking up the table.

C, C_1, C_2, C_3 and C_4 denote the convolution values of the associated gray regions. C can be computed as

$$C = C_1 - C_2 + C_3 - C_4.$$

With the above description of the lookup table based method, we can derive the complexity for sparse simulation method as follows:

Lemma 4 (Sparse simulation method complexity)**.** *The total number of complex number operations using lookup table method is roughly*

$$T_{sparse} = f(K)PL^2, \tag{6.2}$$

where $f(K)$ is written as

$$f(K) = svK^2. \tag{6.3}$$

Remark. *This lemma says that the sparse simulation complexity depends on how sparse (s) the simulations are, the vertex density (v) and the kernel size (K). We will show below that it is more sensitive to the kernel size K than the dense simulation method complexity.*

Proof. Let us consider how many complex number operations are needed to compute the image intensity at a grid point (e.g. O). Because the kernels are of a finite size $K \times K$, only the shapes inside the $K \times K$ region around O need to be considered. The number of vertices per unit area is v. Therefore, the average number vertices in the support region around O is vK^2. There are P kernels. The total number of floating point operations is PvK^2. The total number of grid points is L^2. A fraction (s) of these points is simulated. Therefore, the total intensity simulation needs to be on sL^2 points.

The total number of floating point operations is $sL^2 \times PvK^2 = \underbrace{svK^2}_{f(K)} PL^2$.

□

For dense simulation, the chip of size $L \times L$ is divided into tiles of size $(K' - K) \times (K' - K)$ to save memory usage (Figure 6.6). Each tile is then

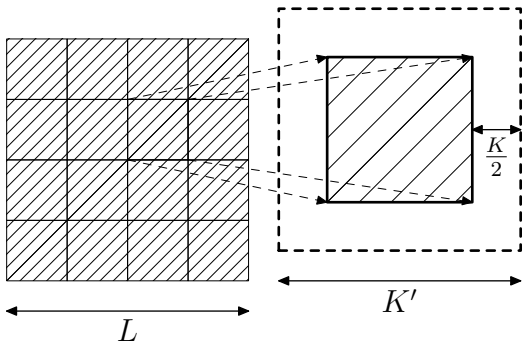


Figure 6.6: A chip with size $L \times L$ is divided into many tiles of size $K' - K$, where K is the same as that of sparse simulation and K' is a tunable parameter. Each tile is zero-padded to of size K'.

zero-padded to of size $K' \times K'$ [85]. The aerial images are computed in each tile and eventually are stitched together. The grid size in the Fourier domain is $\frac{1}{K'\delta/\frac{\lambda}{\mathrm{NA}}}$, where NA is the numerical aperture and λ is the wavelength. The size of the kernel (H_p) area can be written as $K'\mathrm{NA}\frac{\delta}{\lambda}$. The total number of

complex number operations is roughly

$$
\begin{aligned}
T_{\text{dense}} &= \underbrace{\frac{L^2}{(K'-K)^2}}_{\text{no. of tiles}} \Big(\underbrace{CK'^2 \log K'}_{\text{FFT } m} + \underbrace{P}_{\text{no. of kernels}} \\
&\quad \times \Big(\underbrace{(K'\text{NA}\frac{\delta}{\lambda})^2}_{\text{multiply with the kernel } H_p} + \underbrace{CK'^2 \log K'}_{\text{FFT back to the spatial domain}} \Big) \\
&= \frac{L^2}{(K'-K)^2} \Big(CK'^2 \log K' + P\big((K'\text{NA}\frac{\delta}{\lambda})^2 + CK'^2 \log K'\big) \Big) \\
&= L^2 \frac{K'^2}{(K'-K)^2} \Big(C(1+P) \log K' + P(\text{NA}\frac{\delta}{\lambda})^2 \Big) \\
&\underset{\substack{P \gg 1 \\ \text{NA} \to 1}}{\approx} PL^2 \frac{K'^2}{(K'-K)^2} \Big(C \log K' + \big(\frac{\delta}{\lambda}\big)^2 \Big) \\
&\underset{C \log K' \gg (\frac{\delta}{\lambda})^2}{\approx} C \underbrace{\frac{K'^2}{(K'-K)^2} \log K'}_{g(K',K)} PL^2,
\end{aligned} \tag{6.4}
$$

where C is a constant in the order of 1, whose exact value depends on the implementation details of the FFT algorithm [94]. Since we are only interested in a rough estimation of the time complexity, we set NA to 1, and we ignore 1 in $P+1$ since P is usually in the order of 10. These simplifications do not change the order of magnitude of the complexity. For the commonly used lithography systems, the wavelength is around a few hundred nm (e.g., $\lambda = 193\,\text{nm}$). The grid size δ is usually in the order of nm (wafer scale). Thus, we can see the condition of the last approximation in (6.4) ($C \log K' \gg (\frac{\delta}{\lambda})^2$) holds. Therefore, we derive the complexity for dense simulation method as follows:

Lemma 5 (Dense simulation method complexity)**.** *The total number of complex number operations using* (6.1) *is roughly*

$$
T_{dense} = Cg(K', K)PL^2, \tag{6.5}
$$

where

$$g(K', K) = \frac{K'^2}{(K' - K)^2} \log K'. \tag{6.6}$$

The factor PL^2 is common to the runtime complexities of both methods ((6.2) and (6.5)). So we only to need consider the two terms $f(K)$ and $Cg(K', K)$ to compare the complexity of sparse and dense simulation methods.

As the technology node decreases, a bigger kernel size (K) is needed for better model accuracy [16]. It is also shown that etch modeling could also increase the kernel size significantly [38]. Therefore, it is important to study the function $g(K', K)$ when K changes. Because it is not straightforward to show this property of $g(K', K)$ analytically, we show it graphically below.

We plot $g(K', K)$ for different tile size K' and kernel size K in Figure 6.7. It is clear that there is an optimal tile size K' for any given kernel size K.

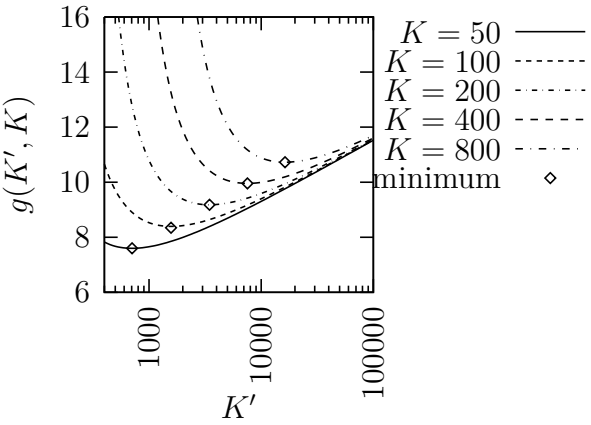


Figure 6.7: Dense simulation complexity factor $g(K', K)$.

Figure 6.8 shows the optimal tile size K' as a function of the kernel size K in a practical range. As we can see that K is almost proportional to

K'_{opt}, which is the optimal value of K'. For the range that is shown, we can approximate K'_{opt} as $K'_{\text{opt}} = 20K$. This conclusion is important for us to fine tune a dense simulator to achieve the fastest speed.

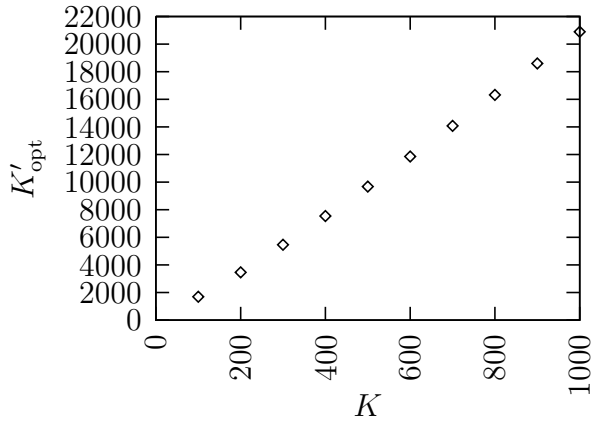


Figure 6.8: Optimal tile size K' as a function of the kernel size K.

Figure 6.9 shows the optimal $Cg(K', K)$ as a function of the kernel size K. We take $C = 4$ for illustration purposes. $s = 1$ means that the

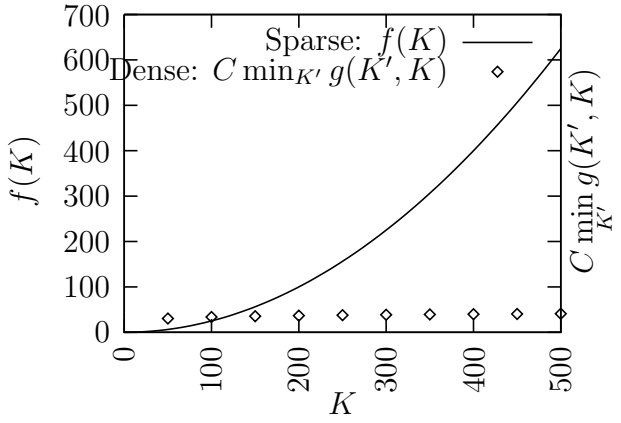


Figure 6.9: Optimal $Cg(K', K)$ as a function of the kernel size K. $C = 4$, $s = 1$ and $v = 1/400$.

simulations are performed at every grid point. v is roughly estimated through

the following example. If we have an via array of size 100 nm, the pitch of the array is 200 nm. If the pixel size on the mask is $\delta = 5$ nm (scaled down to wafer scale), there are $(200/5)^2 = 40^2$ pixels for a via on the average. Each via has 4 vertices. Therefore, $v = 4/40^2 = 1/400$. Figure 6.9 shows that dense simulation method will become better than the sparse simulation method as K grows.

6.3 Topologically Invariant Pixel Base Mask Shape Operations with Lithographic Considerations

Topology is originated from the concept that some geometric problems do not depend on the exact shape of the objects involved, but rather on the way they are put together. Figure 6.10 shows that a mug can be continuously deformed into a donut. These two shapes are topologically equivalent. We can see from this example that two shapes have the same number of continuous regions if they are topologically equivalent. We will formalize such observation for the pixel based mask shapes.

Since both printed shapes and binary mask shapes can be represented as pixel images of two values (0 and 1), we will only study binary images in this work. We will extend the framework for other mask types in the future. We first discuss the connectivity and define the *topological invariance*. Then, we will list all the single pixel mask operations that preserve the shape topology. We will further eliminate the operations which could produce lithography unfriendly shapes. The operations defined here will be used in TIP-OPC.

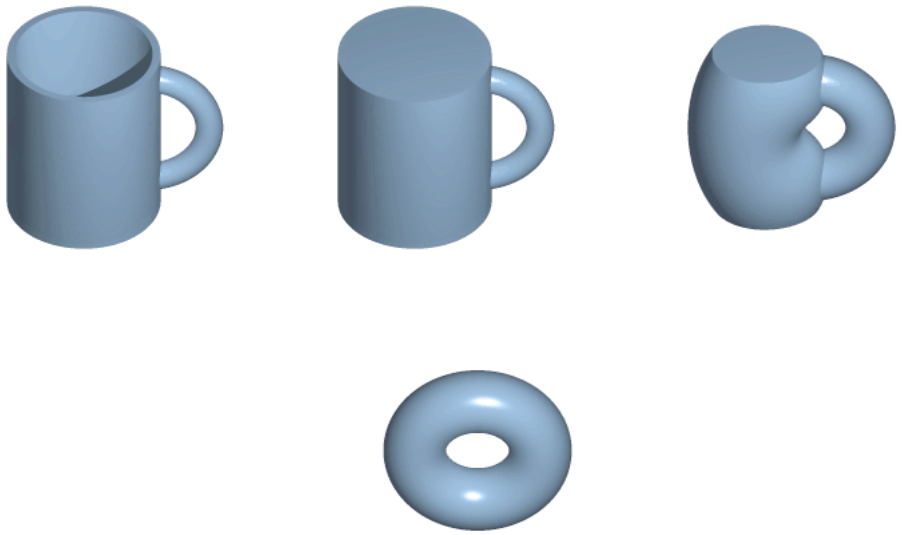


Figure 6.10: A mug is continuously deformed into a donut [95].

6.3.1 Connectivity and Topological Equivalence

Many papers have been published on the thinning of digital images, where some operations are defined to maintain shape topologies [56]. The same mathematical language can be adapted to our problems as below. However, we will modify those operations to fit our specific needs.

A pixel p can be gray (representing value 1) or white (representing value 0). A gray pixel can be turned to white, and a white pixel can be turned to gray. These operations are called flipping-off and flipping-on that pixel.

Definition 1 (Universal set and complement)**.** *A universal set is defined as the set of all the pixels under consideration. For an image of size $m \times n$, the*

universal set can be written as

$$\mathfrak{U} = \{(i,j)|0 \leqslant i < m \text{ and } 0 \leqslant j < n\}. \tag{6.7}$$

Any set of shapes can be denoted as $\mathfrak{M}$*, where*

$$\mathfrak{M} = \{(i,j)|(i,j) \text{ is inside the shapes}\}. \tag{6.8}$$

Its complement $\overline{\mathfrak{M}}$ *is defined as*

$$\overline{\mathfrak{M}} = \mathfrak{U} \setminus \mathfrak{M}, \tag{6.9}$$

where "\" denotes the set subtraction.

Remark. *The universal set shall be understood as the whole mask region in our problem.*

Definition 2 (Neighbor pixels)**.** *As shown in Figure 6.11, the pixels* x_1, x_3, x_5, x_7 *are the 4-neighbors of the pixel* p*. The pixels* $x_1, x_2, \ldots, x_8$ *are the 8-neighbors of* p*.*

x_4	x_3	x_2
x_5	p	x_1
x_6	x_7	x_8

Figure 6.11: Pixels in the neighborhood of p. x_i $(i = 1, 3, 5, 7)$ are 4-neighbors of p. x_i $(i = 1, \ldots, 8)$ are 8-neighbors of p.

Definition 3 (Boundary pixels)**.** *A 4- (or 8-) boundary pixel is a pixel with at least one 4- (or 8-) neighbor pixel having a different color. We call a boundary pixel an* inner boundary *pixel when it is gray, and a boundary pixel an* outer boundary *pixel when it is white. For a gray pixel set* $\mathfrak{M}$*, we denote the set of all its outer boundary pixels as* $\beta_c^+(\mathfrak{M})$ *and the set of all its inner boundary pixels as* $\beta_c^-(\mathfrak{M})$*, where* $c = 4$ *or* 8*. We denote the union of them as* $\beta_c(\mathfrak{M}) = \beta_c^+(\mathfrak{M}) \cup \beta_c^-(\mathfrak{M})$*, called the 4- (or 8-) boundary of* $\mathfrak{M}$*.*

Definition 4 (Path). *A sequence of pixels $y_1, y_2, \ldots, y_n$ is called a 4- (or 8-) path if y_{i+1} is a 4- (or 8-) neighbor of y_i, $i = 1, 2, \ldots, n-1$.*

Definition 5 (Connectivity). *A pair of pixels x, y in a subset $\mathfrak{S}$ of a pixel set $\mathfrak{M}$ are 4- (or 8-) connected if there exists a 4- (or 8-) path from x to y consisting of pixels in $\mathfrak{S}$ only. The subset $\mathfrak{S}$ is 4- (or 8-) connected if every pair of pixels x, y in $\mathfrak{S}$ are connected. In this case, $\mathfrak{S}$ is said to be a 4- (or 8-) component of $\mathfrak{M}$. The number of 4- (or 8-) components of $\mathfrak{M}$ is called the degree of 4- (or 8-) connectivity of $\mathfrak{M}$, denoted as $D_4(\mathfrak{M})$ (or $D_8(\mathfrak{M})$).*

It has been suggested that connectivities for $\mathfrak{S}$ and its complement $\overline{\mathfrak{S}}$ should be different to avoid the paradoxes of $\mathfrak{S}$ and $\overline{\mathfrak{S}}$ being both connected or both disconnected [77]. For example, in Figure 6.12, if the "diamond" loop (consisting of all the gray pixels) is connected, the white pixels should be dissected into two components, intuitively. Otherwise, all the white pixels will be connected. However, if we choose 4-connectivity for both the gray and the

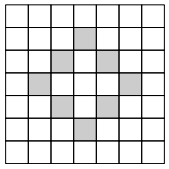


Figure 6.12: The connectivity paradox (taken from [77]).

white pixels, the result is that the pixels on the loop are disconnected and its interior region is also disconnected from its exterior region. If we choose 8-connectivity for both the gray and white pixels, all the gray pixels on the loop are connected and all white pixels are connected. To avoid this paradox, we shall choose different connectivities for gray and white pixels.

In the following discussion, we choose 4-connectivity for gray pixels and 8-connectivity for white pixels, because it is intuitive to require the two mask shapes ($\mathfrak{M}_1$ and $\mathfrak{M}_2$) in Figure 6.13 to be disconnected. We will define *topological equivalence* based on the above choice of connectivities.

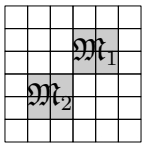


Figure 6.13: 4-connectivity is chosen for gray pixels to make $\mathfrak{M}_1$ and $\mathfrak{M}_2$ disconnected. Therefore, 8-connectivity is chosen for white pixels.

Definition 6 (Topological Equivalence)**.** *Suppose we have two pixel sets $\mathfrak{X}$ and $\mathfrak{Y}$. If we can find a sequence of pixel sets $\mathfrak{M}_1, \ldots, \mathfrak{M}_n$ ($\mathfrak{M}_1 = \mathfrak{X}$ and $\mathfrak{M}_n = \mathfrak{Y}$), which satisfy*

- $\mathfrak{M}_i \neq \mathfrak{M}_{i+1}, \quad i = 1, \ldots, n-1;$
- *there exists some x_i such that*

$$\mathfrak{M}_i \cup x_i = \mathfrak{M}_{i+1} \quad or \quad \mathfrak{M}_i \setminus x_i = \mathfrak{M}_{i+1}, \quad i = 1, \ldots, n-1; \tag{6.10}$$

- *$D_4(\mathfrak{M}_i) = D_4(\mathfrak{M}_j)$ and $D_8(\overline{\mathfrak{M}_i}) = D_8(\overline{\mathfrak{M}_j})$ for any i, j,*

then we can say that $\mathfrak{X}$ and $\mathfrak{Y}$ are topological equivalent.

6.3.2 Topologically Invariant Mask Operations

We define topologically invariant mask operations in this section.

Definition 7 (Removable and insertable)**.** *Suppose $\mathfrak{M}$ is a gray pixel set and its complement $\overline{\mathfrak{M}}$ is a white pixel set. A gray pixel p is removable if flipping it*

off does not change the degree of 4-connectivity of $\mathfrak{M}$, $D_4(\mathfrak{M})$, *and the degree of 8-connectivity of* $\overline{\mathfrak{M}}$, $D_8(\overline{\mathfrak{M}})$. *A white pixel* p *is insertable if flipping it on does not change* $D_4(\mathfrak{M})$ *and* $D_8(\overline{\mathfrak{M}})$.

Remark. *Flipping a removable or insertable pixel at one time does not change mask topology. According to Definition 6, we can sequentially apply these operations without changing the mask topology.*

We will show below which pixels are removable or insertable. Flipping on or off a pixel could only affect the connectivity of its 8-neighbors. Therefore, we only need to examine all the 3×3 pixel patterns (totally $2^{3\times 3} = 512$ patterns).

In Figure 6.14, we show all the removable and insertable cases. In this figure, "x" denotes a pixel that could be gray or white, and "?" is the pixel under consideration. It is easy to check that flipping the "?" pixel for each case does not change the degree of connectivity for gray regions and white regions. These cases under rotation by 90°, 180° and 270° axes are still removable or insertable. Therefore, there is a total of $(1 + 2^2 + 2^3 + 2^4) \times 4 \times 2 = 232$ cases.

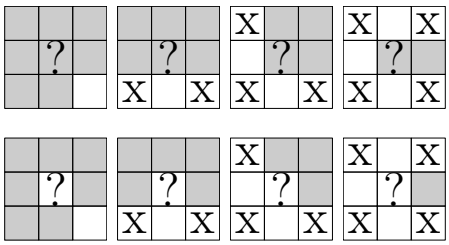


Figure 6.14: Removable (upper row) and insertable (lower row) pixels (denoted by "?"). There are totally 232 cases.

6.3.3 Lithographic Considerations

Because the optical lens is a low-pass filter, the high frequency components associated with fine features cannot pass the lens. Therefore, the sharp corners are not necessary from the lithography imaging point of view. For example, we do not want the patterns in Figure 6.15, since in this representation the size of one pixel is below the resolution of the imaging tool.

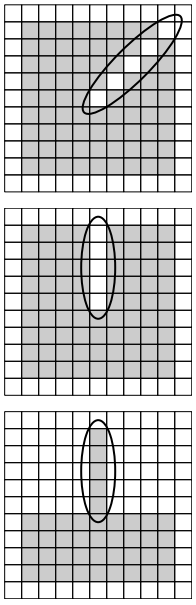


Figure 6.15: Lithography non-friendly features (marked by ellipses).

To circumvent the creation of these three pattens, we do not allow the following three removable or insertable pixels shown in Figure 6.16, respectively. We also do not allow the cases under rotation by 90°, 180° and 270°. Therefore, there a total of $(1 + 1 + 2^2) \times 4 = 24$ cases are not allowed.

Deleting the cases in Figure 6.16 from Figure 6.14 yields removable and insertable pixels that are lithography friendly in Figure 6.17. These cases

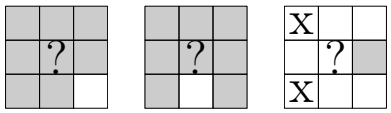


Figure 6.16: Not allowed removable (left and middle) and insertable pixels (right) (denoted by "?") due to lithographic considerations. There are a total of 24 cases.

under rotation by 90°, 180° and 270°, or reflection about x or y axes, are still in this category. Therefore, there are a total of $(((2^2-1)+2^3+2^4)+1+2^2+2^3+(2^4-2^2))\times 4 = 208$ cases. We can check that $208 = 232 - 24$. It is

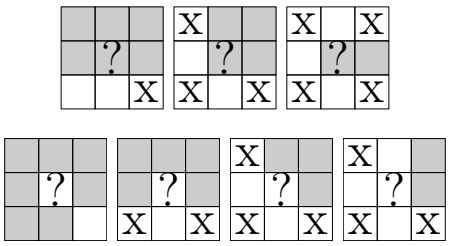


Figure 6.17: Lithography friendly removable (upper row) and insertable (lower row) pixels, marked by "?". There are a total of 208 cases.

easy to confirm that flipping the pixel marked with "?" does not change the topology, and it will not create any fine features like those in Figure 6.15.

Since the center pixels in Figure 6.17 are all boundary pixels, we call them *lithography friendly topologically invariant boundary* pixels. We denote the set of inner and outer boundary pixels as $\gamma^-(\mathfrak{M})$ and $\gamma^+(\mathfrak{M})$, respectively. TIP-OPC (Section 6.4) only flips those pixels. Thus, the mask shape topology is maintained, and we know that no lithography unfriendly pattern is generated.

6.4 Topologically Invariant Pixel Based OPC (TIP-OPC)

In this section, we propose a better contour based cost function than those used in ILT and MB-OPC. Then we show an efficient method to compute the derivative of the cost function with respect to mask changes using FFT. TIP-OPC uses this information to flip the mask pixels which give more improvement on the cost function.

6.4.1 TIP-OPC Cost Function

All the ILT approaches [39, 44, 72, 73] minimize a cost function $F[m]$,

$$\min_{m} F[m], \tag{6.11}$$

where m is the mask transmission function, which can take any value between 0 and 1. The cost function $F[m]$ generally has the following form,

$$F[m] = P_{\mathrm{I}}(I) + P_{\mathrm{c}}(m) + P_{\mathrm{tr}}(m), \tag{6.12}$$

where I denotes the image intensity distribution. The three terms on the right hand side are described below.

The first term $P_{\mathrm{I}}(I)$ in (6.12) penalizes the print image intensity I when it is different from the desired intensity distribution. However, in practice we care about the printed contour, not the intensity distribution. Two different contours might have the same $P_{\mathrm{I}}(I)$, and two different intensity distributions (with different $P_{\mathrm{I}}(I)$) might have the same contours. To eliminate this mismatch between contours and intensity distributions, we explicitly used the contour information, which is the area of the symmetric difference between the printed region R and the target region $\widehat{R}$ (area$(R \triangle \widehat{R})$), in the TIP-OPC cost function. Note that area$(R \triangle \widehat{R}) \geqslant 0$ and area$(R \triangle \widehat{R})$ reduce to zero

if and only if the printed contour R and the target contour $\widehat{R}$ are the same. Since the contour is explicitly expressed in the above cost function, we can expect that its solution is better than the solution from the cost function in (6.12) in terms of contour fidelity.

The second term $P_{\mathrm{c}}(m)$ in (6.12) penalizes complexity masks. Since TIP-OPC maintains the mask shape topology and does not generate any lithography unfriendly patterns, it is possible to do without this term of the TIP-OPC cost function.

The third term $P_{\mathrm{tr}}(m)$ in (6.12) favors masks with certain transmission values, e.g., 0 and 1 for binary masks. TIP-OPC only generates mask pixels with transmission value of 0 and 1. Therefore, this term is not needed.

So we deduce the cost function for TIP-OPC as

$$F[M] \equiv \mathrm{area}(R \triangle \widehat{R}), \tag{6.13}$$

where M denotes the mask shapes. Our new objective (6.13) only concerns with contour fidelity.

The new metric $\mathrm{area}(R \triangle \widehat{R})$ is also better than the Edge Placement Error (EPE) metric, commonly used in MB-OPC algorithm for contour fidelity [109]. EPE denotes the distance between the target contour and the printed contour at the tagging points. Since EPEs are measured sparsely, making them all zero cannot guarantee that the printed contour and the target contour will be the same as shown in Figure 6.18. But $\mathrm{area}(R \triangle \widehat{R})$ is zero if and only if $R = \widehat{R}$.

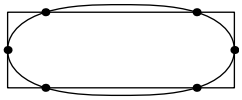


Figure 6.18: EPEs are zero at the tagging points (dots). But the two contours are different.

6.4.2 Efficient Computation of the Cost Sensitivity

We will use the sensitivity of our cost function F with respect to mask changes to guide TIP-OPC. A naive way to compute the sensitivity is to flip each mask pixel, directly compute the new cost function F and check how much F changes. But this is too slow to be used in practice. In the following, we derive a fast way to compute the sensitivity using convolutions, which can be computed by FFT efficiently.

We first derive the sensitivity of F with respect to intensity changes. Since we use a constant threshold photoresist model, a point (x, y) is on the printed contour if and only if

$$I(x, y, m(\cdot, \cdot)) = I_{\text{th}}, \tag{6.14}$$

where $m(\cdot, \cdot)$ is the mask transmission function.

We consider how the contour changes near a contour point (x_0, y_0) due to the changes in the mask transmission function $m(\cdot, \cdot)$. Letting t denote how far the contour point moves, we have

$$I(x_0 + n_x t, y_0 + n_y t, m) = I_{\text{th}}, \tag{6.15}$$

where

$$\boldsymbol{n} = (n_x, n_y) = -\frac{\nabla I}{|\nabla I|} \tag{6.16}$$

is the negative gradient direction of the intensity I, which is perpendicular to the printed contour.

Taking the derivative on both sides of (6.15), we have

$$\frac{\partial I}{\partial x} n_x \delta t + \frac{\partial I}{\partial y} n_y \delta t + \underbrace{\frac{\delta I}{\delta m} \Delta m}_{\delta I} = 0. \tag{6.17}$$

Therefore, using (6.16),

$$\delta t = \frac{\delta I}{\sqrt{\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2}}. \tag{6.18}$$

If we integrate δt on the boundary of R, we get the change of area R, where

$$\delta \operatorname{area}(R) = \oint_{\partial R} \delta t \mathrm{d}s = \oint_{\partial R} \frac{\delta I}{\sqrt{\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2}} \mathrm{d}s. \tag{6.19}$$

Now, let us consider $\delta \operatorname{area}(R \triangle \widehat{R})$. Note that $\widehat{R}$ does not change. Moving the printed contour outward when it is inside $\widehat{R}$ would decrease the intersection area. Otherwise, it would increase the intersection area. Therefore, we have

$$\delta F[M] = \delta \operatorname{area}(R \triangle \widehat{R}) = \oint_{\partial R} \frac{1_{\overline{\widehat{R}}} - 1_{\widehat{R}}}{\sqrt{\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2}} \delta I \mathrm{d}s, \tag{6.20}$$

where $\overline{\widehat{R}}$ is the complement of $\widehat{R}$ and 1_A is the indicator function of the set A defined as

$$\mathbf{1}_A(x) = \begin{cases} 1 & \text{if } x \in A, \\ 0 & \text{if } x \notin A. \end{cases} \tag{6.21}$$

M, R and $\widehat{R}$ in (6.20) are regions not snapped to a grid. Since we need to snap all the regions to a grid during the simulation, we have to represent δF in terms of their corresponding pixel representations, $\mathfrak{M}$, $\mathfrak{R}$ and $\widehat{\mathfrak{R}}$.

A straightforward pixel based version of (6.20) is

$$\delta F[\mathfrak{M}] = \iint_{\beta_4(\mathfrak{R})} \frac{1_{\overline{\widehat{\mathfrak{R}}}} - 1_{\widehat{\mathfrak{R}}}}{\sqrt{\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2}} \delta I \mathrm{d}x\mathrm{d}y, \tag{6.22}$$

where the path integral is replaced by an integral over the print image boundary pixels $\beta_4(\mathfrak{R})$. However, this would not make δF zero even if the print image $\mathfrak{R}$ is the same as the target $\widehat{\mathfrak{R}}$, in which case we would like δF to be zero, since any changes in $\mathfrak{M}$ could make the difference between $\mathfrak{R}$ and $\overline{\mathfrak{R}}$ bigger. Therefore, we modify (6.22) for this case as

$$\delta F[\mathfrak{M}] = \iint_{\beta_4(\mathfrak{R})} \frac{1_{\overline{\widehat{\mathfrak{R}}}\setminus\beta_4^+(\widehat{\mathfrak{R}})} - 1_{\widehat{\mathfrak{R}}\setminus\beta_4^-(\widehat{\mathfrak{R}})}}{\sqrt{\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2}} \delta I \mathrm{d}x\mathrm{d}y. \tag{6.23}$$

Checking $\delta F[\mathfrak{M}]$ in (6.23) shows that it is zero if $\mathfrak{R}$ is the same as $\widehat{\mathfrak{R}}$.

Now, we derive an efficient way of computing the sensitivity $\frac{\delta F}{\delta m}$. From (1.13) we have

$$I[m] = \sum_{p=0}^{2P-1} \sigma_p (h_p * m)^2, \tag{6.24}$$

where σ_p's are coefficients, h_p are kernels and m is the mask. Differentiating I with respect to m and plugging it in (6.23), we can derive

$$\frac{\delta F}{\delta m} = 2 \sum_{p=0}^{2P-1} \sigma_p \left(\frac{1_{\beta_4(\mathfrak{R})}(1_{\overline{\widehat{\mathfrak{R}}}\setminus\beta_4^+(\widehat{\mathfrak{R}})} - 1_{\widehat{\mathfrak{R}}\setminus\beta_4^-(\widehat{\mathfrak{R}})})(h_p * m)}{\sqrt{\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2}} * P h_p \right), \tag{6.25}$$

where P is an operator defined as

$$Pw(x, y) = w(-x, -y), \tag{6.26}$$

for any two argument function $w(x, y)$. The convolutions $h_p * m$'s are the intermediate results of the intensity computation through the dense simulation

method (6.1), which do not incur additional runtime since we compute the intensity anyway. Since (6.25) consists of convolutions, we can still compute them for all the mask pixels using FFT efficiently. Using the naive method mentioned at the beginning of this subsection, we would have to flip each mask pixel, resimulate the intensity, check the intensity changes and check the cost change. Resimulating the intensity for each pixel change is very expensive, but our FFT based method is apparently much faster than that. TIP-OPC uses this sensitivity information to effectively reduce the cost function F.

6.4.3 The Overall TIP-OPC algorithm

The TIP-OPC algorithm (Algorithm 9) is an iterative algorithm which successively modifies the mask $\mathfrak{M}$ using the mask operations defined in Section 6.3 to reduce the cost function $F[\mathfrak{M}]$. The algorithm stops if no improvement can be made in the cost function.

Since TIP-OPC flips on mask pixels in the topologically invariant outer boundary $\gamma^+(\mathfrak{M})$ and flips off mask pixels in the topological invariant outer boundary $\gamma^-(\mathfrak{M})$, we only need $\frac{\delta F}{\delta m}$ on $\gamma^-(\mathfrak{M}) \cup \gamma^+(\mathfrak{M})$. It is easy to see the following facts based on the definition of $\frac{\delta F}{\delta m}$ and the fact that only the two mask transmission values 0 and 1 are allowed:

- A gray pixel can be flipped off if it has positive value of $\frac{\delta F}{\delta m}$.
- A white pixel can be flipped on if it has negative value of $\frac{\delta F}{\delta m}$.

We call the absolute value $\frac{\delta F}{\delta m}$ in the above two cases as the *useful* value of $\frac{\delta F}{\delta m}$. Line 5 in Algorithm 9 computes the maximum useful $\frac{\delta F}{\delta m}$ (denoted as a in the algorithm). The algorithm modifies the pixels whose useful $\frac{\delta F}{\delta m}$ are between

Algorithm 9 TIP-OPC algorithm

1: **function** TIPOPC($\widehat{\mathfrak{R}}$)
2: $\mathfrak{M} \leftarrow \widehat{\mathfrak{R}}$ // initialize mask to target
3: **repeat**
4: compute $\frac{\delta F}{\delta m}$
5: $a \leftarrow \max\Big(\max\limits_{p \in \gamma^-(\mathfrak{M})} \frac{\delta F}{\delta m}, -\min\limits_{p \in \gamma^+(\mathfrak{M})} \frac{\delta F}{\delta m}\Big)$
6: **if** $a \leqslant 0$ **then**
7: **break**
8: $b \leftarrow a/2$
9: `failed` $\leftarrow$ **true**
10: **for** $i \leftarrow 1, n$ **do**
11: $\mathfrak{M}' \leftarrow$ AdjustMask($\mathfrak{M}, a, b$)
12: **if** cost is reduced **then**
13: `failed` $\leftarrow$ **false**
14: $\mathfrak{M} \leftarrow \mathfrak{M}'$
15: **break**
16: **else**
17: $b \leftarrow b/2$
18: **until** `failed`

$(a/2, a]$, $(3a/4, a]$, $(7a/8, a]$, ..., until the cost function improves or fails even after n trials. In the later case, the algorithm stops.

Algorithm 10 shows how the mask is adjusted to maintain the topology invariance. Note that $\mathfrak{M}'$ and $\mathfrak{M}$ have the same topology at the beginning. Each change in Line 6 maintains $\mathfrak{M}'$'s topology. Therefore, at the end of the program, $\mathfrak{M}'$ is still topologically equivalent to the input $\mathfrak{M}$. Therefore, the topology of $\mathfrak{M}$ is maintained.

Algorithm 10 Mask Adjustment algorithm

```
1: function AdjustMask(𝔐, a, b)
2:     𝔐′ ← 𝔐 // make a copy
3:     for pixel p ∈ γ⁺(𝔐) ∪ γ⁻(𝔐) do
4:         if p's useful δF/δm is in (a − b, a] then
5:             if flip p on 𝔐′ if it does not change 𝔐′'s topology then
6:                 flip p on M′
7:     return 𝔐′
```

6.5 Results of Experiments

We implemented both the sparse and dense lithography simulators, the current commonly used MB-OPC algorithm described in [23], and the TIP-OPC algorithm described in this chapter using C++. The grid size of $\delta = 5\,\text{nm}$ was used for the following experiments.

Based on the lithography friendly topologically invariant mask operations, no SRAFs and excessively small features are created. Therefore, we know that TIP-OPC generates less complex masks than ILT by theory. Therefore, we do not need to compare them experimentally.

6.5.1 Runtime Comparison of Sparse and Dense Simulations

We used both sparse and dense sparse simulation methods on a 114×114 via array to simulate the aerial image on all the grid points. Each via was of size $100\,\text{nm} \times 100\,\text{nm}$. The pitches in both x- and y-directions were 200 nm. We took $K = 240$ and $K' = 20 \times K = 4800$. We employed $P = 6$ kernels in the simulations.

The runtime for sparse and dense simulation methods were 1.99×10^3 sec and 70.8 sec. We can see that the sparse simulation method can be much slower than the dense simulation method for dense simulation applications. Note that our estimation of sparse simulation runtime is optimistic in (6.2), because we ignore the processing time of the polygon shapes.

6.5.2 Quality Comparison of MB-OPC and TIP-OPC

We used quadrupole illumination with illumination parameters $\sigma_{\text{center}} = 0.85$ and $\sigma_{\text{radius}} = 0.2$. The numerical aperture was NA $= 0.8$, the wavelength was $\lambda = 193\,\text{nm}$ and the intensity threshold was $I_{\text{th}} = 0.1$. We employed $P = 8$ kernels in the simulations. We took $n = 10$ in TIP-OPC. For MB-OPC, we used the segment length of 100 nm.

The MB-OPC and TIP-OPC results of 6 test patterns across 130 nm, 90 nm, and 65 nm technology nodes are shown in Table 6.1. These patterns are typical poly and metal 1 patterns. We used area$(R \triangle \widehat{R})$ as the metric for the contour fidelity. The "total" columns denote area$(\widehat{R})$ in the unit of pixel. The errors are in the unit of pixel, as well. The "ratio" columns denote the relative error area$(R \triangle \widehat{R})$/ area$(\widehat{R})$. The "Reduction" columns show the reduction in the relative error between MB-OPC and TIP-OPC. The average reductions are shown for each technology generation. The table shows that as technology

scales down, TIP-OPC improves in terms of relative error reduction. Thus, it may be needed in future generations when the error budget become more stringent.

Note that as technology scales down, the relative error in TIP-OPC also goes up. This does not mean TIP-OPC does not do a good job. No OPC can make corners print perfectly since the lithography optical system cannot pass the high frequency components associated with the corners. As the technology scales down, the percent of the corner region increases. Therefore, we see an increase in the relative error for TIP-OPC, as well. The errors reflect the fact that no other OPC algorithm can correct those residue errors, which means that the layouts should be modified to be more OPC friendly if we want the error be reduced further.

In Figure 6.19, we show the OPCed mask for the pattern "pat2" (five-jog pattern). As we can see TIP-OPC maintains topologies and does not create any SRAFs. TIP-OPC also does not create any lithography unfriendly mask features. Higher contour fidelity and low mask complexity demonstrate that TIP-OPC is indeed a promising method.

6.5.3 Combining TIP-OPC with SRAF insertion

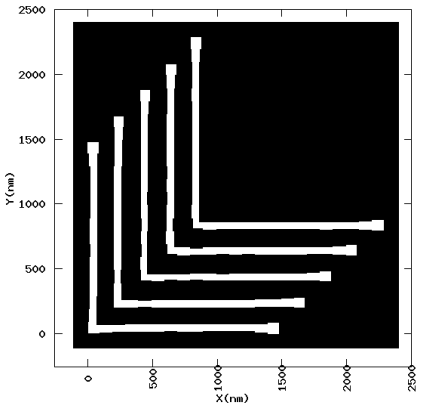
TIP-OPC does not insert SRAFs directly. But it can be used with other SRAFs insertion algorithms, such as rule based method [52] and interference mapping lithography (IML) [86]. We apply TIP-OPC to a poly layout on which SRAFs have been inserted based on rules. Figure 6.20 shows the resulting mask. Here, we use the same lithography settings as those of the previous section.

Table 6.1: OPC quality comparison

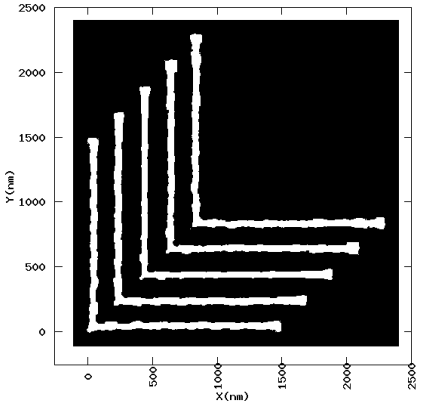
130nm						
pattern	total	MB-OPC		TIP-OPC		Reduction
		error	ratio	error	ratio	
pat1	48216	3780	7.84%	1053	2.18%	5.66%
pat2	227200	15615	6.87%	1914	0.84%	6.03%
pat3	38130	2833	7.43%	720	1.89%	5.54%
pat4	53360	4021	7.54%	1177	2.21%	5.33%
pat5	42255	3172	7.51%	799	1.89%	5.62%
pat6	58860	4493	7.64%	1225	2.08%	5.55%
average						5.62%

90nm						
pattern	total	MB-OPC		TIP-OPC		Reduction
		error	ratio	error	ratio	
pat1	24360	2126	8.73%	797	3.27%	5.46%
pat2	113680	7113	6.26%	1806	1.59%	4.67%
pat3	19533	2796	14.31%	1176	6.02%	8.29%
pat4	27294	3647	13.36%	1136	4.16%	9.20%
pat5	21714	2934	13.51%	1161	5.35%	8.17%
pat6	30252	4088	13.51%	2023	6.69%	6.83%
average						7.10%

65nm						
pattern	total	MB-OPC		TIP-OPC		Reduction
		error	ratio	error	ratio	
pat1	12600	1609	12.77%	788	6.25%	6.52%
pat2	58000	5176	8.92%	2091	3.61%	5.32%
pat3	10026	3618	36.09%	2466	24.60%	11.49%
pat4	13968	5314	38.04%	2642	18.91%	19.13%
pat5	11118	3813	34.30%	2480	22.31%	11.99%
pat6	15424	5703	36.97%	3322	21.54%	15.44%
average						11.65%



(a) MB-OPC



(b) TIP-OPC

Figure 6.19: OPCed mask of the pattern “pat2”.

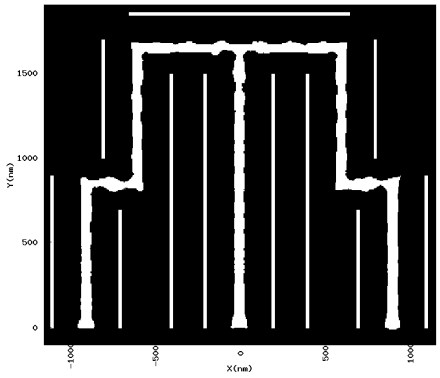


Figure 6.20: TIP-OPC with SRAF insertion.

6.6 Summary

In this chapter, we proposed the topologically invariant pixel based OPC (TIP-OPC) paradigm. As dense simulations become widely adopted as feature sizes shrink to deep sub-wavelength, using pixel based mask representation is inevitable. To reduce mask complexity that could result from ILT, the current pixel based approach, we maintain the mask shape topology and do not allow lithography unfriendly mask patterns. Our TIP-OPC algorithm has efficient lithography friendly topologically invariant mask operations and FFT based cost-to-mask sensitivity computation. Our experimental results show that TIP-OPC is better than MB-OPC in terms of contour fidelity. Since the mask shapes are topologically invariant, the resulting mask has relatively low mask complexity compared with ILT.

Chapter 7

Conclusions

In this dissertation, we have improved the accuracy and speed of lithography simulation. We use an iterative integration method to improve the accuracy of TCC computation. We implement the new TCC computation algorithm in a new software package ELIAS that can be extended easily to handle various lithography simulation conditions. We speed up the conventional image simulation method based OCAs by about 2× using the symmetric properties of the lithography imaging system. We also provide a variational lithography model (VLIM) that models variations in lithography systems.

In the aspect of OPC algorithms, we developed a process variation aware OPC algorithm based VLIM to make corrected lithography patterns more robust with respect to process variations. We speed up the conventional OPC algorithm by making it intensity based, which saves runtime by reducing the total amount of lithography image intensity simulations. Besides the above improvements on vector based OPC algorithms, we proposed the topologically invariant pixel based paradigm for pixel-based OPC (TIP-OPC), which reduces mask complexity compared with other pixel based algorithms and increases contour fidelity compared with vector based algorithms.

Future study should extend ELIAS so that it can handle cases where the illumination functions are continuous but have large derivatives over certain areas. In practice, the analytical formula of an illumination function is

usually not available, and the illumination function has to be measured and saved as an intensity map. Determining how to measure the map and integrate it with ELIAS is also a future research topic. There are other topics that will become increasingly important, such as the 3D mask effect, source-mask optimization, extreme ultraviolet (EUV) lithography simulation and the corresponding OPC. I believe that there will be many research opportunities in design-for-manufacturing (DFM) for future generations of semiconductor manufacturing technologies.

Appendices

Appendix A

Proofs of Theorems in Chapter 2

Theorem 1. We use D to denote the support of $u(x, y)$ and ∂D to denote the boundary of the support. In addition, we denote the bounds of $u(x, y)$ and its first and second derivatives in R as

$$|u| \leqslant \zeta, \tag{A.1}$$

$$\left|\frac{\partial u}{\partial x}\right| \leqslant \eta \quad \text{and} \quad \left|\frac{\partial u}{\partial y}\right| \leqslant \eta, \tag{A.2}$$

$$\left|\frac{\partial^2 u}{\partial x^2}\right| \leqslant \theta \quad \text{and} \quad \left|\frac{\partial^2 u}{\partial y^2}\right| \leqslant \theta, \tag{A.3}$$

where ζ, η and θ are all constants.

The truncation error of $M_{R,\Delta}(u)$ (see (2.5)) can be written as

$$\begin{aligned} |I_R(u) - M_{R,\Delta}(u)| \leqslant & \sum_{ij}^{(0)} |I_{\square_{ij}}(u) - \Delta^2 u(\boxdot_{ij})| \\ & + \sum_{ij}^{(1)} |I_{\square_{ij}}(u) - \Delta^2 u(\boxdot_{ij})|, \end{aligned} \tag{A.4}$$

where the superscripts of $\sum$'s indicate how many functions in the integrand are discontinuous and $I_{\square}(u)$ denotes the integration of u over the square $\square$

$$I_{\square}(u) = \iint_{\square} u(x, y)\, \mathrm{d}x\mathrm{d}y.$$

The first term on the right hand side of (A.4) is summed over all squares where u is smooth, and the second term is summed over all squares where u is discontinuous.

The truncation error for each square can be described as in the following two cases.

1. The function $u(x,y)$ is smooth in the square $\square$. According to the Taylor's theorem,

$$\begin{aligned} u(x,y) =& u(x_0,y_0) \\ &+ \Big((x-x_0)\frac{\partial}{\partial x} + (y-y_0)\frac{\partial}{\partial y}\Big)u(x_0,y_0) \\ &+ \frac{1}{2}\Big((x-x_0)\frac{\partial}{\partial x} + (y-y_0)\frac{\partial}{\partial y}\Big)^2 u(x^*,y^*), \end{aligned}$$

where (x_0,y_0) is the square center $\boxdot$, and (x^*,y^*) is a point satisfying

$$(x^*-x_0, y^*-y_0) = (\lambda(x-x_0), \lambda(y-y_0)), \quad 0<\lambda<1.$$

Therefore, we have the truncation error

$$\begin{aligned} & |I_\square(u) - \Delta^2 u(x_0,y_0)| \\ \leqslant & \Big|\iint_\square (u(x,y)-u(x_0,y_0))\,\mathrm{d}x\mathrm{d}y\Big| \\ = & \iint_\square \frac{1}{2}\Big((x-x_0)^2\frac{\partial^2}{\partial x^2} + (y-y_0)^2\frac{\partial^2}{\partial y^2}\Big) \\ & \qquad \times u(x^*,y^*)\,\mathrm{d}x\mathrm{d}y \\ \leqslant & \frac{\theta}{2}\iint_\square ((x-x_0)^2+(y-y_0)^2)\,\mathrm{d}x\mathrm{d}y = \frac{\theta}{12}\Delta^4. \end{aligned} \tag{A.5}$$

2. The function $u(x,y)$ is discontinuous in the square $\square$. The truncation error can be estimated as

$$\begin{aligned} |I_\square(u) - \Delta^2 u(x_0,y_0)| &\leqslant \iint_\square |u(x,y)-u(x_0,y_0)|\,\mathrm{d}x\mathrm{d}y \\ &\leqslant \iint_\square \zeta\,\mathrm{d}x\mathrm{d}y = \zeta\Delta^2. \end{aligned} \tag{A.6}$$

By using (A.4), (A.5) and (A.6), we can derive that the truncation error of $M_{R,\Delta}(u)$ is bounded as follows:

$$|I_R(u) - M_{R,\Delta}(u)| \leqslant \frac{\theta}{12}\sum_{ij}^{(0)}\Delta^4 + \zeta\sum_{ij}^{(1)}\Delta^2.$$

As an example, we show the support of a function in Figure A.1. The

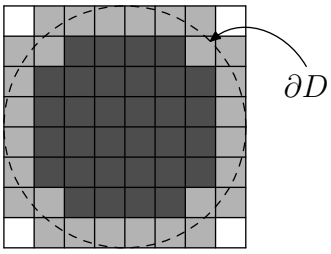


Figure A.1: The support D of the function is the region enclosed by the circle which is denoted as ∂D. The summation $\sum_{ij}^{(0)}$ is over the dark gray squares, and the summation $\sum_{ij}^{(1)}$ is over the light gray squares.

summations $\sum_{ij}^{(0)}$ and $\sum_{ij}^{(1)}$ are indicated by the dark gray squares and the light gray squares. It is obvious that the number of dark gray squares is bounded by $A(D)/\Delta^2$, where $A(D)$ is the total area of the support D, and the number of light gray squares is bounded by $C_{\mathrm{l}}l(\partial D)/\Delta$, where C_{l} is a constant and $l(\partial D)$ is the length of the boundary ∂D. Therefore, the truncation error can be estimated as

$$|I_R(u) - M_{R,\Delta}(u)| \leqslant C_1\Delta^2 + C_2\Delta, \tag{A.7}$$

where $C_1 = A(D)\theta/12$ and $C_2 = C_l l(\partial D)\zeta$. □

Theorem 2. According to Algorithm 1, $M_{\square,\Delta'}(u)$ can be written as

$$\begin{aligned} M_{\square,\Delta'}(u) &= \sum_k \Delta_k^2 \sum_i u(\boxdot_{k,i}) \\ &= \sum_{ki}^{(0)} \Delta_k^2 u(\boxdot_{k,i}) + \sum_i^{(1)} \Delta_0^2 u(\boxdot_{0,i}), \end{aligned}$$

where Δ_k denotes the square size and the subscript $_{k,i}$ is the index of a not-divided square of size Δ_k. The superscript of the summation sign again denotes whether u is smooth or discontinuous. Therefore, we have the truncation error

$$\begin{aligned} &|I_\square(u) - M_{\square,\Delta'}(u)| \\ \leqslant &\sum_{ki}^{(0)} |I_{\square_{k,i}}(u) - \Delta_k^2 u(\boxdot_{k,i})| + \sum_i^{(1)} |I_{\square_{0,i}}(u) - \Delta_0^2 u(\boxdot_{0,i})| \\ \leqslant &\sum_{ki}^{(0)} \frac{\theta}{12} \Delta_k^4 + \sum_i^{(1)} \zeta \Delta_0^2. \end{aligned}$$

Using the inequality

$$\sum_i x_i^2 \leqslant \big(\sum_i x_i\big)^2, \quad \text{for } x_i \geqslant 0,$$

we have

$$\sum_{ki}^{(0)} \Delta_k^4 \leqslant \big(\sum_{ki}^{(0)} \Delta_k^2\big)^2 \leqslant \big(\sum_{ki} \Delta_k^2\big)^2 \leqslant (\Delta^2)^2 = \Delta^4.$$

Similar to Proof A, the number of terms in the summation $\sum_i^{(1)}$ is bounded by $\frac{C_l l_\square(D)}{\Delta_0}$, where C_l is the constant that we mentioned in Proof A and $l_\square(\partial D)$ is the length of the boundary ∂D in the square $\square$. Therefore, we have

$$\begin{aligned} |I_\square(u) - M_{\square,\Delta'}(u)| &\leqslant \frac{\theta}{12} \Delta^4 + \frac{C_\mathrm{l} l_\square(\partial D)}{\Delta_0} \zeta \Delta_0^2 \\ &\leqslant \frac{\theta}{12} \Delta^4 + C_\mathrm{l} l_\square(\partial D) \zeta \Delta' \end{aligned} \tag{A.8}$$

We can then derive that the truncation error of $M_{R,\Delta,\Delta'}(u)$ is bounded as

$$|I_R(u) - M_{R,\Delta,\Delta'}(u)| \leqslant C_1\Delta^2 + C_2\Delta', \tag{A.9}$$

where $C_1 = \frac{A(D)\theta}{12}$ and $C_2 = C_l l(\partial D)\zeta$. □

Theorem 3. Let T_i be the runtime of Algorithm 1 for a square of the size Δ_i. Because Algorithm 1 is a recursive algorithm, we can approximate T_i by a recursive sequence

$$T_{i+1} = C_\mathrm{d} T_i + (4 - C_\mathrm{d})T_0 = C_\mathrm{d} T_i + b,$$

where C_d denote the number of the smaller squares of size Δ_i that needs to be further divided, and $b = (4 - C_\mathrm{d})T_0$, where T_0 is the runtime of the midpoint rule for a square that is not divided.

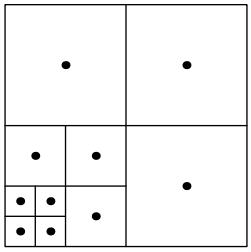
The constant C_d is bounded ($1 \leqslant C_\mathrm{d} < 4$) practically:

1. Figure A.2(a) shows the case for $C_\mathrm{d} = 1$, which rarely happens in practice. The recursion will not be sustained if C_d is smaller than 1.

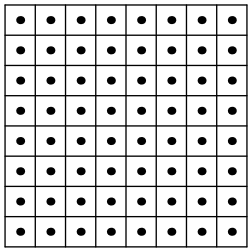
2. $C_\mathrm{d} = 4$ means that each square is quadrisected unless it is small enough. This is equivalent to use a uniform grid as in Figure A.2(b), which is impossible, because cutting all the squares to a small enough size means that the curve is not simple.

We approximate C_d for different i's by a same constant C_d ($0 < C_\mathrm{d} < 2$), which is some kind of "average" over all i's. We can transform the recursive relation to

$$T_{i+1} - \beta = C_\mathrm{d}(T_i - \beta),$$



(a) $C_{\mathrm{d}} = 1$



(b) $C_{\mathrm{d}} = 4$

Figure A.2: Two extreme cases of the recursive quadrisection of a square.

where $\beta = \frac{b}{1-C_{\mathrm{d}}}$. Therefore,

$$T_n = C_{\mathrm{d}}^n(T_0 - \beta) + \beta = \alpha C_{\mathrm{d}}^n + \beta,$$

where $\alpha = T_0 - \beta$. The level of recursion n can be approximated as $n = \log \frac{\Delta}{\Delta'}$. Therefore, the runtime of the recursive integration (2.8) over a square $\square$ of size Δ is

$$T_{\square,\Delta'} = \alpha C_{\mathrm{d}}^{\log \frac{\Delta}{\Delta'}} = \alpha \Big(\frac{\Delta}{\Delta'}\Big)^{\log C_{\mathrm{d}}}, \tag{A.10}$$

where the additive constant β is ignored for large $\frac{\Delta}{\Delta'}$ and $0 < \log C_{\mathrm{d}} < 2$. $\square$

(Theorem 4). By distinguishing whether $u(x,y)$ $v(x,y)$ $w(x,y)$ are discontin-

uous or not, we can approximate $I_R(uvw)$ as

$$\begin{aligned}
&M_{R,\Delta,\Delta'}(u,v,w)\\
=&\Delta^2\sum_{ij}^{(0)} u(\boxdot_{ij})v(\boxdot_{ij})w(\boxdot_{ij}) + \sum_{ij}^{(1)} M_{\square_{ij},\Delta'}(u,vw)\\
&+\sum_{ij}^{(1)} M_{\square_{ij},\Delta'}(v,uw) + \sum_{ij}^{(1)} M_{\square_{ij},\Delta'}(w,uv)\\
&+\sum_{ij}^{(2)} M_{\square_{ij},\Delta'}(vw,u) + \sum_{ij}^{(2)} M_{\square_{ij},\Delta'}(uw,v)\\
&+\sum_{ij}^{(2)} M_{\square_{ij},\Delta'}(uv,w) + \sum_{ij}^{(3)} M_{\square_{ij},\Delta'}(uvw),
\end{aligned} \tag{A.11}$$

where the superscripts $^{(n)}$ $(n = 0, 1, 2, 3)$ of the summation signs denote the number of functions of u, v and w that are discontinuous in $\square_{ij}$, and the discontinuous functions are in the left arguments of $M_{\square_{ij},\Delta'}(\cdot,\cdot)$.

The truncation error of (2.14) can be written as

$$\begin{aligned}
&|I_\square(uv) - M_{\square,\Delta'}(u,v)|\\
=&\left|\iint_\square u(x,y)v(x,y)\,\mathrm{d}x\mathrm{d}y - \iint_\square \frac{M_{\square,\Delta'}(u)}{\Delta^2}v(x_0,y_0)\,\mathrm{d}x\mathrm{d}y\right|\\
\leqslant&\underbrace{\left|\iint_\square \Big(u(x,y) - \frac{M_{\square,\Delta'}(u)}{\Delta^2}\Big)v(x,y)\,\mathrm{d}x\mathrm{d}y\right|}_{\text{see (A.12)}}\\
&+\underbrace{\left|\iint_\square \frac{M_{\square,\Delta'}(u)}{\Delta^2}(v(x,y) - v(x_0,y_0))\,\mathrm{d}x\mathrm{d}y\right|}_{\text{see (A.13)}}.
\end{aligned}$$

Using Taylor's theorem, we have

$$\begin{aligned}
&\left| \iint_{\square} \Big(u(x,y) - \frac{M_{\square,\Delta'}(u)}{\Delta^2}\Big) v(x,y)\ \mathrm{d}x\mathrm{d}y \right| \\
\leqslant &\left| \iint_{\square} \Big(u(x,y) - \frac{M_{\square,\Delta'}(u)}{\Delta^2}\Big) v(x_0,y_0)\ \mathrm{d}x\mathrm{d}y \right| \\
&+ \left| \iint_{\square} \Big(u(x,y) - \frac{M_{\square,\Delta'}(u)}{\Delta^2}\Big) \right. \\
&\qquad \left. \times \Big((x-x_i)\frac{\partial}{\partial x} + (y-y_i)\frac{\partial}{\partial y}\Big) v(x^*,y^*)\ \mathrm{d}x\mathrm{d}y \right| \\
\leqslant &\zeta_v |I_{\square}(u) - M_{\square,\Delta'}(u)| + \zeta_u \eta_v \iint_{\square} (|x-x_i| + |y-y_i|)\ \mathrm{d}x\mathrm{d}y \\
\leqslant &\zeta_v \Big(\frac{\theta_u}{12}\Delta^4 + C l_{\square}(D_u)\zeta_u \Delta'\Big) + \zeta_u \eta_v \frac{\Delta^3}{2},
\end{aligned} \tag{A.12}$$

where we have used (A.8). Here, ζ, η and θ are still the bounds of functions and their derivatives, and their subscripts denote what the functions are. We also have

$$\begin{aligned}
&\left| \iint_{\square} \frac{M_{\square,\Delta'}(u)}{\Delta^2} (v(x,y) - v(x_0,y_0))\ \mathrm{d}x\mathrm{d}y \right| \\
\leqslant &\zeta_u |I_{\square}(v) - M_{\square}(v)| = \frac{\zeta_u \theta_v}{12}\Delta^4,
\end{aligned} \tag{A.13}$$

where we have used (A.5). Ignoring (A.13), which is bounded by a higher order term of Δ, the truncation error of (2.14) is bounded as

$$|I_{\square}(uv) - M_{\square,\Delta'}(u,v)| \leqslant C l_{\square}(D_u)\zeta_u\zeta_v\Delta' + \zeta_u\eta_v\frac{\Delta^3}{2}. \tag{A.14}$$

Let $f(x,y) = u(x,y)v(x,y)w(x,y)$, and we have

$$I_R(f) = \sum_{ij}^{(0)} I_{\square_{ij}}(f) + \sum_{ij}^{(1)} I_{\square_{ij}}(f). \tag{A.15}$$

It is clear that the number of terms in the summations $\sum_{ij}^{(1)}$, $\sum_{ij}^{(2)}$ and $\sum_{ij}^{(3)}$ in (A.11) is the same as the number of terms in the summation $\sum_{ij}^{(1)}$ of (A.15),

which is bounded by $C_l l(\partial D)/\Delta$. Here, D is the support of the function $u(x,y)v(x,y)w(x,y)$.

Using (A.5), (A.14) and (A.8), the truncation error of (A.11) can be estimated as

$$\begin{aligned}&|I_{\square}(uvw) - M_{R,\Delta,\Delta'}(u,v,w)|\\ \leqslant&\frac{A(D)}{\Delta^2}C_1\Delta^4 + C_{\mathrm{l}} l(\partial D)C_2\Delta' + \frac{C_{\mathrm{l}} l(\partial D)}{\Delta}C_3\Delta^3,\end{aligned}$$

where C_1, C_2 and C_3 are constants depending on the bounds on the functions and their first and second order derivatives. Therefore,

$$\begin{aligned}&|I_{\square}(uvw) - M_{R,\Delta,\Delta'}(u,v,w)|\\ \leqslant&(A(D)C_1 + C_{\mathrm{l}} l(\partial D)C_3)\Delta^2 + C_{\mathrm{l}} l_R(D_{uvw})C_2\Delta',\end{aligned} \tag{A.16}$$

which is of the same order as (A.9).

Noting that $M_{\square,\Delta'}(u)$ is the same as $\Delta^2 u(\boxdot)$, if $u(x,y)$ is smooth in the square $\square$, we can easily derive (2.15) with some simple mathematical manipulations. $\square$

Appendix B

Proofs of Theorems in Chapter 3

In this section, we will prove that the eigenfunctions a real Hermitian operator under certain conditions can be made either symmetric or antisymmetric. This result will be applied to our lithography image simulation problem at the end of this appendix.

Define the operator A based on a real function $A(\boldsymbol{k}, \boldsymbol{k}')$ as

$$A\phi(\boldsymbol{k}) = \int A(\boldsymbol{k}, \boldsymbol{k}')\phi(\boldsymbol{k}')\mathrm{d}\boldsymbol{k}',$$

where $A(\boldsymbol{k}, \boldsymbol{k}') = A(\boldsymbol{k}', \boldsymbol{k})$. Define the parity operator P as

$$P\phi(\boldsymbol{k}) = \phi(-\boldsymbol{k}). \tag{B.1}$$

Theorem 6. *The parity operator P has only eigenvalues 1 and -1. If ψ is the eigenfunction associated with the eigenvalue 1, then $\psi(\boldsymbol{k}) = \psi(-\boldsymbol{k})$. If ψ is the eigenfunction associated with the eigenvalue -1, then $\psi(\boldsymbol{k}) = -\psi(-\boldsymbol{k})$.*

Proof. Assume ψ is an eigenfunction of P such that $P\psi = \lambda\psi$.

We put

$$\psi_1(\boldsymbol{k}) = \frac{\psi(\boldsymbol{k}) + \psi(-\boldsymbol{k})}{2},$$
$$\psi_2(\boldsymbol{k}) = \frac{\psi(\boldsymbol{k}) - \psi(-\boldsymbol{k})}{2}.$$

Then $P\psi = P\psi_1 + P\psi_2 = \psi_1 - \psi_2$. Since $P\psi = \lambda\psi$, we have $\psi_1 - \psi_2 = \lambda\psi_1 + \lambda\psi_2$. Then we have $(\lambda - 1)\psi_1 = (\lambda + 1)\psi_2$.

Since ψ_1 is an even function and ψ_2 is an odd function, we have $\lambda = 1$ or $\lambda = -1$. So we can say that the parity operator P has only eigenvalues 1 and -1.

Obviously, if $\lambda = 1$, then $\psi_2 = 0$. So we have $\psi(\boldsymbol{k}) = \psi(-\boldsymbol{k})$. If $\lambda = -1$, then $\psi_1 = 0$. So we have $\psi(\boldsymbol{k}) = -\psi(-\boldsymbol{k})$. □

Theorem 7. *If $A(\boldsymbol{k}, \boldsymbol{k}')$ is real and $A(\boldsymbol{k}, \boldsymbol{k}') = A(-\boldsymbol{k}', -\boldsymbol{k})$, $A(\boldsymbol{k}, \boldsymbol{k}')$ can be expanded in terms of orthonormal real functions $\psi_i(\boldsymbol{k})$ as*

$$A(\boldsymbol{k}, \boldsymbol{k}') = \sum_{i=1}^{\infty} \sigma_i \psi_i(\boldsymbol{k}) \psi_i(\boldsymbol{k}'),$$

where $\psi_i(\boldsymbol{k})$ satisfies

$$\psi_i(\boldsymbol{k}) = \psi_i(-\boldsymbol{k}) \text{ or } \psi_i(\boldsymbol{k}) = -\psi_i(-\boldsymbol{k}).$$

Proof. Assume $\{\sigma_i\}$ are eigenvalues and $\{\phi_i\}$ are normalized eigenfucntions.

Since A is real and symmetric, σ_i and ϕ_i are real with $\int \phi_i(\boldsymbol{k})\phi_j(\boldsymbol{k})\mathrm{d}\boldsymbol{k} = \delta_{ij}$ for any $i, j \in \mathbb{N}$.

It is easy to see that

$$\int A(\boldsymbol{k}, \boldsymbol{k}')\phi_i(-\boldsymbol{k}')\mathrm{d}\boldsymbol{k}' = \sigma_i \phi_i(-\boldsymbol{k}).$$

This implies that

$$\sigma_i(P\phi_i) = A(P\phi_i),$$

for any $i \in \mathbb{N}$.

So $P\phi_i$ is still an eigenfunction of A associated with engenvalue σ_i.

Since A is a compact operator from L_2 to L_2, then $\forall\lambda \in \mathbb{R}$, if $\lambda \neq 0$, there are at most finitely many $i \in \mathbb{N}$ such that $\sigma_i = \lambda$.

Without losing generality, we assume $\sigma_1 = \cdots = \sigma_n \neq \sigma_{n+1}$.

Put $V = \text{span}\{\phi_1, \cdots, \phi_n\}$, then $P(V) \subset V$. So there is an orthonormal basis $\{\psi_1, \cdots, \psi_n\}$ of V such that $P\psi_i = \psi_i$ or $P\psi_i = -\psi_i$ for $1 \leq i \leq n$.

It is easy to see that

$$\begin{aligned}&\phi_1(\boldsymbol{k})\phi_1(\boldsymbol{k}') + \cdots + \phi_n(\boldsymbol{k})\phi_n(\boldsymbol{k}')\\ =&\psi_1(\boldsymbol{k})\psi_1(\boldsymbol{k}') + \cdots + \psi_n(\boldsymbol{k})\psi_n(\boldsymbol{k}').\end{aligned}$$

Then we can conclude that

$$A(\boldsymbol{k}, \boldsymbol{k}') = \sum_{i=1}^{\infty} \sigma_i \psi_i(\boldsymbol{k})\psi_i(\boldsymbol{k}'),$$

with $\psi_i(\boldsymbol{k}) = \psi_i(-\boldsymbol{k})$ or $\psi_i(\boldsymbol{k}) = -\psi_i(-\boldsymbol{k})$ for $i \in \mathbb{N}$ and $\psi_i(\boldsymbol{k})$ is real. □

Since $\mathcal{T}_{\text{real}}$ satisfies the conditions of Theorem 7, we have

$$\mathcal{T}_{\text{real}}(\boldsymbol{k}', \boldsymbol{k}'') = \sum_n \sigma_n \mathcal{Q}_n(\boldsymbol{k}')\mathcal{Q}_n(\boldsymbol{k}''), \tag{B.2}$$

where each σ_n is a real number, $\mathcal{Q}_n(\boldsymbol{k})$ is real, and $\mathcal{Q}_n(\boldsymbol{k})$ satisfies

$$\mathcal{Q}_n(\boldsymbol{k}) = \mathcal{Q}_n(-\boldsymbol{k}) \text{ or } \mathcal{Q}_n(\boldsymbol{k}) = -\mathcal{Q}_n(-\boldsymbol{k}). \tag{B.3}$$

Therefore, Eq. (3.20) can be derived [67], where Q_n is the inverse Fourier transform of $\mathcal{Q}_n$.

Using Eq. (B.3), it is easy to see that the inverse Fourier transform Q_n of $\mathcal{Q}_n$ is either real or purely imaginary. So the magnitude operator $|\cdot|$ in Eq. (3.22) is not necessary, because even it is pure imaginary, we can make it real by multiplying the imaginary unit i, $iQ_n \to Q_n$. Therefore, taking only p' terms as an approximation, we can prove Eq. (3.26).

Appendix C

VLIM Based Bossung Curves Deduction

Focus-Exposure Matrix is defined as the variation of line width (and possibly other parameters) as a function of both focus and exposure energy. The data is typically plotted as line width versus focus for different exposure energies and these plots are often referred to as smiley plots, spider plots, or Bossung curves [63].

Before we show how to reduce the model data into curves which have analytical formulas, we discuss the requirement on the production lithography systems; that is, maturity. This means process condition variations should be small enough such that photoresist profile changes should also be small. There should not be anything like photoresist collapse, etc. Otherwise, if the process is too sensitive to process condition variations or the process condition variations are too large, we should improve the process first rather than using any tricks to survive in that process.

Using this assumption, the line width (or CD) can be expressed as polynomials of exposure dose and focus error. Based on this idea, various polynomial fitting functions have been proposed [9, 32, 62]. In VLIM, we have diffusion length parameter d. We assume CD can also be expressed as a polynomial of d. By taking the $z \leftrightarrow -z$ symmetry, we have only even order

terms of z. The CD function can be expressed as

$$\mathrm{CD}^{\boldsymbol{P}}_{\mathrm{VLIM}}(I_{\mathrm{th}}, z, d) = \sum_{l=0}^{L}\sum_{m=0}^{M}\sum_{n=0}^{N} a_{lmn} I_{\mathrm{th}}^{l} z^{2m} d^{n}, \tag{C.1}$$

where the bias B is set as 0 in this case. a_{lmn} and I_{th0} are fitting parameters. The superscript $^{\boldsymbol{P}}$ denotes that the different CD function for different pattern $\boldsymbol{P}$. Using non-linear least square regression, we can fit the CD function to VLIM to CD data generated at various I_{th}, z, d. The upper limits, L, M and N, of the summations are chosen heuristically. They should not too large or too small to overfit or underfit the curves.

Bibliography

[1] D. S. Abrams and L. Pang. Fast inverse lithography technology. In *Proc. SPIE 6154*, pages 534–542, April 2006.

[2] K. Adam, Y. Granik, A. Torres, and N. B. Cobb. Improved modeling performance with an adapted vectorial formulation of the Hopkins imaging equation. In *Proc. SPIE 5040*, pages 78–91, June 2003.

[3] Konstantinos Adam and Wilhelm Maurer. Polarization effects in immersion lithography. *Journal of Micro/Nanolithography, MEMS and MOEMS*, 4(3):031106, 2005.

[4] C.-N. Ahn, H.-B. Kim, and K.-H. Baik. Novel approximate model for resist process. In *Proc. SPIE 3334*, pages 752–763, June 1998.

[5] M. Al-Imam and W. A. Tawfic. Optimization of OPC runtime using efficient optical simulation. In *Proc. SPIE 6730*, volume 6730, October 2007.

[6] Charles Albertalli and Tom Kingsley. Computational Lithography (cover story). *Semiconductor International*, 30(5):36–42, May 2007.

[7] Max Born and Emil Wolf. *Principles of Optics : Electromagnetic Theory of Propagation, Interference and Diffraction of Light*. Cambridge University Press, 7 edition, October 1999.

[8] A. Borna, C. Progler, and D. Blaauw. Correlation analysis of CD-variation and circuit performance under multiple sources of variability. In *Proc. SPIE 5756*, pages 168–177, May 2005.

[9] A. Bourov, S. A. Robertson, B. W. Smith, M. Slocum, and E. C. Piscani. Experimental measurement of photoresist modulation curves. In *Proc. SPIE 6154*, pages 1003–1008, April 2006.

[10] T. A. Brunner and R. A. Ferguson. Approximate models for resist processing effects. In *Proc. SPIE 2726*, pages 198–207, June 1996.

[11] T. A. Brunner, C. Fonseca, N. Seong, and M. Burkhardt. Impact of resist blur on MEF, OPC, and CD control. In *Proc. SPIE 5377*, pages 141–149, May 2004.

[12] Timothy A. Brunner. Why optical lithography will live forever. *Journal of Vacuum Science & Technology B: Microelectronics and Nanometer Structures*, 21(6):2632–2637, 2003.

[13] B.B. Chaudhuri and S. Chandrashekhar. Neighboring direction run-length coding: an efficient contour codingscheme. *IEEE Transactions on Systems, Man and Cybernetics*, 20(4):916–921, 1990.

[14] Chun-Kuang Chen, Tsai-Sheng Gau, Jaw-Jung Shin, Ru-Gun Liu, Shinn-Sheng Yu, Anthony Yen, and Burn J. Lin. Mask error tensor and causality of mask error enhancement for low-k_1 imaging: theory and experiments. *J. Microlithogr. Microfabrication, Microsyst.*, 3(2):269–275, April 2004.

[15] N. Cobb. Flexible sparse and dense OPC algorithms. In *Proc. SPIE 5853*, pages 693–702, June 2005.

[16] N. Cobb and D. Dudau. Dense OPC and verification for 45nm. In *Proc. SPIE 6154*, pages 191–196, April 2006.

[17] N. Cobb and Y. Granik. New concepts in OPC. In *Proc. SPIE 5377*, pages 680–690, 2004.

[18] N. B. Cobb and Y. Granik. OPC methods to improve image slope and process window. In *Proc. SPIE 5042*, pages 116–125, 2003.

[19] N. B. Cobb and Y. Granik. Dense OPC for 65nm and below. In *Proc. SPIE 5992*, pages 1521–1532, November 2005.

[20] N. B. Cobb and A. Zakhor. Fast, low-complexity mask design. In *Proc. SPIE 2440*, pages 313–327, May 1995.

[21] N. B. Cobb, A. Zakhor, and E. Miloslavsky. Mathematical and CAD framework for proximity correction. In *Proc. SPIE 2726*, pages 208–222, June 1996.

[22] Nicolas B. Cobb and Yuri Granik. Model-based opc using the meef matrix. In *Proc. SPIE 4889*, pages 1281–1292, 2002.

[23] Nicolas B. Cobb and Yuri Granik. Model-based OPC using the MEEF matrix. In *Proc. SPIE 4889*, pages 1281–1292, December 2002.

[24] Nicolas Bailey Cobb. *Fast Optical and Process Proximity Correction Algorithms for Integrated Circuit Manufacturing*. PhD thesis, University of California at Berkeley, 1998.

[25] Daniel C. Cole, Eytan Barouch, Edward W. Conrad, and Michael Yeung. Using Advanced Simulation to Aid Microlithography Development. *Proc. IEEE*, 89(8):1194–1213, 2001.

[26] Edward W. Conrad, Daniel C. Cole, David P. Paul, and Eytan Barouch. Model considerations, calibration issues, and metrology methods for resist-bias models. In *Proc. SPIE 3677*, pages 940–955, 1999.

[27] A. D. Dave, C. P. Babcock, S. N. McGowan, and Y. Zou. Methods and factors to optimize OPC run-time. In *Proc. SPIE 6520*, 2007.

[28] Philip J. Davis and Philip Rabinowitz. *Methods of Numerical Integration*, chapter 2. Academic Press, 2 edition, June 1984.

[29] M. Do, J. Kang, J. Choi, J. Lee, Y. Lee, and K. Kim. Efficient approach to improving pattern fidelity with multi-OPC model and recipe. In *Proc. SPIE 6349*, October 2006.

[30] C. Dolainsky and W. Maurer. Application of a simple resist model to fast optical proximity correction. In *Proc. SPIE 3051*, pages 774–780, July 1997.

[31] P.D. Flanner, III. Two-dimensional optical imaging for photolithography simulation. Technical Report UCB/ERL M86/57, EECS Department, University of California, Berkeley, 1986.

[32] D. Fuard, M. Besacier, and P. Schiavone. Assessment of different simplified resist models. In *Proc. SPIE 4691*, pages 1266–1277, 2002.

[33] D. Fuard, M. Besacier, and P. Schiavone. Validity of the diffused aerial image model: an assessment based on multiple test cases. In *Proc. SPIE 5040*, pages 1536–1543, 2003.

[34] J. G. Garofalo, J. Demarco, J. Bailey, J. Xiao, and S. Vaidya. Reduction of ASIC gate-level line-end shortening by mask compensation. In *Proc. SPIE 2440*, pages 171–183, May 1995.

[35] George A. Gomba. Collaborative Innovation: IBM's Immersion Lithography Strategy for 65 nm and 45 nm Half-pitch Nodes & Beyond. In *Proc. SPIE 6521*, 2007.

[36] George A. Gomba. Collaborative Innovation: IBM's Immersion Lithography Strategy for 65 nm and 45 nm Half-pitch Nodes & Beyond. In *Proc. SPIE 6521*, 2007.

[37] R. L. Gordon. Exact computation of scalar 2D aerial imagery. In *Proc. SPIE 4692*, pages 517–528, July 2002.

[38] Y. Granik. Dry etch proximity modeling in mask fabrication. In *Proc. SPIE 5130*, pages 86–91, August 2003.

[39] Y. Granik. Solving inverse problems of optical microlithography. In *Proc. SPIE 5754*, pages 506–526, May 2005.

[40] Y. Granik and N. B. Cobb. MEEF as a matrix. In *Proc. SPIE 4562*, pages 980–991, 2002.

[41] Y. Granik, N. B. Cobb, and T. Do. Universal process modeling with VTRE for OPC. In *Proc. SPIE 4691*, pages 377–394, July 2002.

[42] Y. Granik, D. M. Medvedev, and N. Cobb. Toward standard process models for OPC. In *Proc. SPIE 6520*, April 2007.

[43] Yuri Granik. Generalized mask error enhancement factor theory. *Journal of Microlithography, Microfabrication, and Microsystems*, 4(2):023001, 2005.

[44] Yuri Granik. Fast pixel-based mask optimization for inverse lithography. *Journal of Microlithography, Microfabrication, and Microsystems*, 5(4):043002, December 2006.

[45] T. Graves, M. D. Smith, and C. A. Mack. Methods for benchmarking photolithography simulators: part IV. In *Proc. SPIE 6154*, pages 982–992, April 2006.

[46] P. Gupta, A. B. Kahng, C.-H. Park, K. Samadi, and X. Xu. Wafer topography-aware optical proximity correction for better DOF margin and CD control. In *Proc. SPIE 5853*, pages 844–854, 2005.

[47] Puneet Gupta and Fook-Luen Heng. Toward a systematic-variation aware timing methodology. In *Proc. Design Automation Conference*, pages 321–326, 2004.

[48] Puneet Gupta, Andrew B. Kahng, Dennis Sylvester, and Jie Yang. Performance Driven OPC for Mask Cost Reduction. In *Proc. Int. Symp. on Quality Electronic Design*, pages 270–275, 2005.

[49] H. H. Hopkins. On the Diffraction Theory of Optical Images. In *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, volume 217, pages 408–432, May 1953.

[50] W. C. Huang, C. M. Lai, B. Luo, C. K. Tsai, M. H. Chih, C. W. Lai, C. C. Kuo, R. G. Liu, and H. T. Lin. Intelligent model-based OPC. In *Proc. SPIE 6154*, pages 1065–1073, April 2006.

[51] C.-Y. Hung, B. Zhang, D. Tang, E. Guo, L. Pang, Y. Liu, A. Moore, and K. Wang. First 65nm tape-out using inverse lithography technology (ILT). In *Proc. SPIE 5992*, pages 596–604, November 2005.

[52] Lawrence S. Melvin III, Jeffrey P. Mayhew, Benjamin D. Painter, and Levi D. Barnes. Assist feature placement analysis using focus sensitivity models. In *Proc. SPIE 6281*, number 1, page 62810X, 2006.

[53] T. Kaneko and M. Okudaira. Encoding of Arbitrary Curves Based on the Chain Code Representation. *IEEE Transactions on Communications*, 33(7):697–707, July 1985.

[54] R. Köhle. Fast TCC algorithm for the model building of high NA lithography simulation. In *Proc. SPIE 5754*, pages 918–929, May 2004.

[55] A. Krasnoperova, J. A. Culp, I. Graur, S. Mansfield, M. Al-Imam, and H. Maaty. Process window OPC for reduced process variability and enhanced yield. In *Proc. SPIE 6154*, pages 1200–1211, April 2006.

[56] Louisa Lam, Seong-Whan Lee, and Ching Y. Suen. Thinning methodologies-a comprehensive survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14(9):869–885, 1992.

[57] L. Liebmann, S. Mansfield, G. Han, J. Culp, J. Hibbeler, and R. Tsai. Reducing DfM to practice: the lithography manufacturability assessor. In *Proc. SPIE 6156*, pages 178–189, April 2006.

[58] Burn J. Lin. Where is the lost resolution. In *Proc. SPIE 633*, page 44, 1986.

[59] Burn J. Lin. The k_3 coefficient in nonparaxial lambda/na scaling equations for resolution, depth of focus, and immersion lithography. *Journal of Microlithography, Microfabrication, and Microsystems*, 1(1):7–12, 2002.

[60] Y. Liu, D. Abrams, L. Pang, and A. Moore. Inverse lithography technology principles in practice: unintuitive patterns. In *Proc. SPIE 5992*, pages 886–893, November 2005.

[61] C.-C. Lu and J.G. Dunham. Highly efficient coding schemes for contour lines based on chain code representations. *IEEE Transactions on Communications*, 39(10):1511–1514, October 1991.

[62] C. A. Mack and J. D. Byers. New model for focus-exposure data analysis. In *Proc. SPIE 5038*, pages 396–405, May 2003.

[63] Chris A. Mack. *Inside PROLITH: A Comprehensive Guide to Optical Lithography Simulation*. Finle Technologies Inc., 1997.

[64] Joydeep Mitra, Peng Yu, and David Z. Pan. RADAR: RET-aware detailed routing using fast lithography simulations. In *Proc. DAC*, pages 369–372, 2005.

[65] B. Painter, L. L. Melvin, III, and M. L. Rieger. Classical control theory applied to OPC correction segment convergence. In *Proc. SPIE 5377*, pages 1198–1206, May 2004.

[66] L. Pang, Y. Liu, and D. Abrams. Inverse Lithography Technology (ILT): what is the impact to the photomask industry? In *Proc. SPIE 6283*, June 2006.

[67] Y. C. Pati and T. Kailath. Phase-shifting masks for microlithography: automated design and mask requirements. *Journal of the Optical Society of America A*, 11:2438–2452, September 1994.

[68] Y.C. Pati, A.A. Ghazanfarian, and R.F. Pease. Exploiting structure in fast aerial image computation for integrated circuit patterns. *IEEE Trans. on Semiconductor Manufacturing*, 10(1):62–74, February 1997.

[69] David M. Pawlowski, Liang Deng, and Martin D. F. Wong. Fast and Accurate OPC for Standard-Cell Layouts. In *Proc. Asia and South Pacific Design Automation Conf.*, pages 7–12, January 2007.

[70] Weifeng Qiu Peng Yu and David Z. Pan. Fast Lithography Image Simulation By Exploiting Symmetries in Lithography Systems. *IEEE Trans. on Semiconductor Manufacturing*, 21(4):638–645, May 2008.

[71] Wilhelm M. Pieper. Recursive Multidimensional Integration. *Int. J. Numer. Methods Eng.*, 40(10):1923–1935, 1997.

[72] A. Poonawala and P. Milanfar. OPC and PSM design using inverse lithography: a nonlinear optimization approach. In *Proc. SPIE 6154*, pages 1159–1172, April 2006.

[73] A. Poonawala and P. Milanfar. Mask Design For Optical Microlithography — An Inverse Imaging Problem. *IEEE Trans. on Image Processing*, 16(3):774–788, March 2007.

[74] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, New York, NY, USA, 2 edition, 1992.

[75] C. J. Progler, A. Borna, D. Blaauw, and P. Sixt. Impact of lithography variability on statistical timing behavior. In *Proc. SPIE 5379*, pages 101–110, 2004.

[76] J. Randall, K. Ronse, T. Marschner, M. Goethals, and M. Ercken. Variable-threshold resist models for lithography simulation. In *Proc. SPIE 3679*, pages 176–182, July 1999.

[77] Azriel Rosenfeld and John L. Pfaltz. Sequential operations in digital picture processing. *J. ACM*, 13(4):471–494, 1966.

[78] J. Schacht, K. Herold, R. Zimmermann, J. A. Torres, W. Maurer, Y. Granik, C.-H. Chang, G. K.-C. Hung, and B. S.-M. Lin. Calibration of OPC models for multiple focus conditions. In *Proc. SPIE 5377*, pages 691–702, May 2004.

[79] Sean X. Shi, Peng Yu, and David Z. Pan. A Unified Non-Rectangular Device and Circuit Simulation Model for Timing and Power. In *Proc. ICCAD*, 2006.

[80] M. D. Smith, J. D. Byers, and C. A. Mack. Comparison between the process windows calculated with full and simplified resist models. In *Proc. SPIE 4691*, pages 1199–1210, July 2002.

[81] M. D. Smith, J. D. Byers, and C. A. Mack. Methods for benchmarking photolithography simulators: part II. In *Proc. SPIE 5377*, pages 1475–1486, May 2004.

[82] M. D. Smith, T. Graves, J. D. Byers, and C. A. Mack. Methods for benchmarking photolithography simulators: Part III. In *Proc. SPIE 5992*, pages 1595–1603, November 2005.

[83] M. D. Smith and C. A. Mack. Methods for benchmarking photolithography simulators. In *Proc. SPIE 5040*, pages 57–68, June 2003.

[84] M. D. Smith and C. A. Mack. Process sensitivity and optimization with full and simplified resist models. In *Proc. SPIE 5040*, pages 1509–1520, June 2003.

[85] Steven W. Smith. *The Scientist & Engineer's Guide to Digital Signal Processing*, chapter 18. California Technical Pub., 1 edition, 1997.

[86] R. J. Socha, D. J. Van Den Broeke, S. D. Hsu, J. F. Chen, T. L. Laidig, N. Corcoran, U. Hollerbach, K. E. Wampler, X. Shi, and W. Conley. Contact hole reticle optimization by using interference mapping lithography (IML). In *Proc. SPIE 5377*, pages 222–240, May 2004.

[87] J. L. Sturtevant, J. Word, P. LaCour, J. W. Park, and D. Smith. Assessing the impact of real world manufacturing lithography variations on post-OPC CD control. In *Proc. SPIE 5756*, pages 240–254, 2005.

[88] John L. Sturtevant, J. A. Torres, J. Word, Y. Granik, and P. LaCour. Considerations for the use of defocus models for OPC. In *Proc. SPIE 5756*, pages 427–436, 2005.

[89] B. Tollkühn, M. Uhle, J. Fuhrmann, K. Gärtner, A. Heubner, and A. Erdmann. Benchmark of a lithography simulation tool for next generation applications. *Microelectronic Engineering*, 83(4-9):1142–1147, 2006.

[90] J. A. Torres, T. Roessler, and Y. Granik. Process window modeling using compact models. In *Proc. SPIE 5567*, pages 638–648, October 2004.

[91] D. Van Den Broeke, M. Hsu, J. F. Chen, S. Hsu, U. Hollerbach, and T. Laidig. Manufacturing implementation of IML^{TM} technology for 45nm node contact masks. In *Proc. SPIE 6283*, June 2006.

[92] J. van der Gracht. Simulation of partially coherent imaging by outer-product expansion. *Applied Optics*, 33(17):3725–3731, June 1994.

[93] Yao-Ting Wang, Y. C. Pati, and Thomas Kailath. Depth of focus and the moment expansion. *OPTICS LETTERS*, 20(18):1841–1843, September 1995.

[94] Wikipedia. Fast Fourier transform — Wikipedia, The Free Encyclopedia, 2007.

[95] Wikipedia. Topology — Wikipedia, The Free Encyclopedia, 2008.

[96] Jim Wiley. Future challenges in computational lithography. *Solid State Technology*, 49(5):68, May 2006.

[97] Alfred K. Wong. Microlithography: Trends, challenges, solutions, and their impact on design. *IEEE Micro*, 23(2):12–21, 2003.

[98] Alfred Kwok-Kit Wong. *Resolution Enhancement Techniques in Optical Lithography*. SPIE Publications, March 2001.

[99] Alfred Kwok-Kit Wong. *Optical Imaging in Projection Microlithography*, volume TT66 of *SPIE Tutorial Texts in Optical Engineering*. SPIE Publications, March 2005.

[100] Alfred Kwok-Kit Wong. *Optical Imaging in Projection Microlithography*. SPIE Publications, March 2005.

[101] Jie Yang, Luigi Capodieci, and Dennis Sylvester. Advanced timing analysis based on post-opc extraction of critical dimensions. In *Proc. Design Automation Conf.*, pages 359–364, 2005.

[102] Jun Ye, Yen-Wen Lu, Yu Cao, Luoqi Chen, and Xun Chen. System and method for lithography simulation.

[103] J.-Y. Yoo, Y.-K. Kwon, J.-T. Park, D.-S. Sohn, S.-G. Kim, Y.-S. Sohn, and H.-K. Oh. CD prediction by threshold energy resist model (TERM). In *Proc. SPIE 4691*, pages 1287–1295, July 2002.

[104] P. Yu, D. Z. Pan, and C. A. Mack. Fast lithography simulation under focus variations for OPC and layout optimizations. In *Proc. SPIE 6156*, pages 397–406, April 2006.

[105] Peng Yu. ELIAS. `http://www.cerc.utexas.edu/utda/download/download.html`.

[106] Peng Yu and David Z. Pan. A Novel Intensity Based Optical Proximity Correction Algorithm with Speedup in Lithography Simulation. In *Proc. Int. Conf. on Computer Aided Design*, pages 854–859, 2007.

[107] Peng Yu and David Z. Pan. Tip-opc: a new topological invariant paradigm for pixel based optical proximity correction. In *Proc. Int. Conf. on Computer Aided Design*, pages 847–853, 2007.

[108] Peng Yu and David Z. Pan. ELIAS: An Accurate and Extensible Lithography Aerial Image Simulator with Improved Numerical Algorithms. *IEEE Trans. on Semiconductor Manufacturing*, May 2009.

[109] Peng Yu, Sean X. Shi, and David Z. Pan. Process Variation Aware OPC with Variational Lithography Modeling. In *Proc. Design Automation Conf.*, pages 785–790, 2006.

[110] Peng Yu, Sean X. Shi, and David Z. Pan. True Process Variation Aware Optical Proximity Correction with Variational Lithography Modeling and Model Calibration. *Journal of Micro/Nanolithography, MEMS and MOEMS*, 6(3):031004, July–September 2007.

Vita

Peng Yu received the B.S. degree in physics from Peking University, China, in 2002, and the M.S. degree in physics from the University of California, San Diego, in 2004. He is currently working toward the Ph.D. degree in the Department of Electrical and Computer Engineering, The University of Texas at Austin. He has interned at Synopsys, IBM, and Cadence. His research interests include design for manufacturing, OPC algorithms, and lithography modeling. He has had several publications in *DAC*, *ICCAD*, *SPIE Microlithography*, and *IEEE Transactions on Semiconductor Manufacturing*. Mr. Yu has reviewed papers for the IEEE Transactions on Computer Aided DESIGN and various conferences. He is a student member of SPIE. He has received the DAC Young Student Support Program Award, IBM Ph.D. Fellowship nomination, BACUS Photomask Scholarship from SPIE, The University of Texas Graduate School Continuing Fellowship, and Inventor Recognition Award from Semiconductor Research Corporation.

Permanent address: 302 W 38TH ST APT 208
Austin, Texas 78705

This dissertation was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.